VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MODERNÁ HRA BLOCKSTACKER

A MODERN BLOCKSTACKER GAME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MAREK DANČO**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MILET TOMÁŠ, Ph.D.**
SUPERVISOR

BRNO 2023

Abstrakt

Táto práca sa zaoberá vytvorením programovateľnej puzzle hry. Hra funguje na princípe padania náhodne generovaných dielikov a odstraňovania vyplnených riadkov podľa svojej hlavnej inšpirácie Tetris. Mimo jednoduchý základ hra podporuje veľké množstvo nastavení, ktoré upravujú ovládanie, menia výzor, ozvučenie, alebo dokonca aj pravidlá hry. Časti hry sú programovateľné vďaka Lua rozhraniu, ktoré je do hry vstavané pomocou knižnice NLua. Celá práca bola vypracovaná v hernom engine Unity.

Abstract

This thesis deals with the creation of a programmable puzzle game. The game works on the principle of randomly generated falling pieces and clearing of full lines, as inspired by Tetris. Notable additions to the game are a large amount of settings, which can alter the controls, appearance, sound or even the rules of the game. Parts of the game are programmable thanks to the Lua interface that is built into the game with the NLua library. The game has been developed using the Unity game engine.

Kľúčové slová

skladanie, puzzle, Tetris, Unity, hra, C#, Lua, skriptovateľné, programovateľné

Keywords

blocks, stacker, Tetris, puzzle, Unity, game, C#, Lua, scriptable, programmable

Citácia

DANČO, Marek. *Moderná hra Blockstacker*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Milet Tomáš, Ph.D.

Moderná hra Blockstacker

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána doktora Mileta. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Marek Dančo
8. mája 2023

Poďakovanie

Chcel by som poďakovať vedúcemu práce doktorovi Tomášovi Miletovi za vedenie práce, rady a konzultácie. Ďalej by som chcel poďakovať kamarátom, ktorí mi s prácou pomáhali - Marekovi Farkašovi za grafický dizajn, Radoslavovi Žilkovi za audio dizajn, Alexandrovi Vránovi za nápady. Ďalej chcem poďakovať všetkým, ktorí pomáhali s testovaním.

Obsah

Kapitola 1

Úvod

Cieľom tejto práce bolo vytvoriť užívateľsky príjemnú, zábavnú hru. Veľkou inšpiráciou bola hra Tetris[1], ktorá je známa ako jedna z najpredávanejších hier na svete. Dnes existuje niekoľko oficiálnych hier, ktoré používajú rovnaké princípy a ešte viac fanúšikovských hier, ktoré sú častokrát aj populárnejšie.

Žáner, ktorý vznikol hrou Tetris v roku 1984 a ktorý je dnes známy ako blockstacker hry, je dnes pomerne kultová záležitosť, napriek nedávnej obrovskej popularite. Veľká väčšina ľudí vie len o existencii klasických blockstacker hier, ku ktorým sa radia napríklad Tetris pre NES, v ktorom sa dodnes hrávajú majstrovstvá sveta.

Už je to ale dávno, čo blockstacker hry nie sú len jednoduché padanie dielikov a snaženie sa nevytvoriť si diery v stavbe. Časom sa pridávali do hier nové a nové mechaniky, medzi ktoré patria na príklad:

- automatické pohybovanie pri držaní tlačidiel,
- videnie niekoľko dielikov dopredu,
- odloženie si dielika na neskôr,
- rôzne spôsoby náhodného generovania dielikov, ktoré hru robia jednoduchšou,
- rotačné systémy, ktoré dovoľujú hráčom v rámci otočenia dieliku aj dielik premiestniť,
- rôzne spôsoby počítania skóre.

Okrem toho boli zavedené nastavenia, ktoré dokážu značne zjednodušiť ovládanie hry. Oficiálne aj fanúšikovské hry predstavili rôzne herné módy, ktoré poskytujú rôzne spôsoby výzvy pre hráčov, v ktorých sa dodnes pomerne často dosahujú svetové rekordy.

Mimo zlepšení v módoch pre jedného hráča boli takisto vytvorené niekoľké spôsoby ako umožniť hrať blockstacker hry viacerým ľuďom proti sebe. Najznámejšie z týchto hier sú oficiálne Tetris hry, ako Puyo Puyo Tetris[2] od spoločnosti Sega, Tetris 99[3] od spoločnosti Nintendo a Tetris Effect: Connected od spoločnosti Enhance[4]. Najobľúbenejšie sú však

[1] https://tetris.com/
[2] https://puyo.sega.com/tetris/
[3] https://www.nintendo.com/store/products/tetris-99-switch/
[4] https://www.tetriseffect.game/

dnes medzi hráčmi fanúšikovské hry ako TETR.IO[5] a Jstris[6] vďaka väčším možnostiam ovládania, lepšiemu užívateľskému rozhraniu a prívetivejšej monetizácii.

Cieľom tejto bakalárskej práce je vytvoriť blockstacker hru, ktorá bude schopná podporovať čo najviac herných mechaník, či už prevzatých z iných hier, alebo originálnych. Takisto je cieľom umožniť hráčom si hru prispôsobiť poskytnutím nastavení, ktoré ovládajú výzor, ozvučenie a správanie hry.

Hra je dostupná na stránke https://mrakdun-desu.itch.io/ustacker.

Nasledujúce kapitoly obsahujú:

- podrobnejší popis blockstacker hier, ich mechaník a vybraných moderných blockstacker hier v kapitole 2,
- popis funkcionality tejto bakalárskej práce v kapitole 3,
- popis použitých technológií v kapitole 4,
- opis implementácie v kapitole 5,
- zhodnotenie priebehu práce a výsledku v kapitole 6.

[5] https://ch.tetr.io/
[6] https://jstris.jezevec10.com/

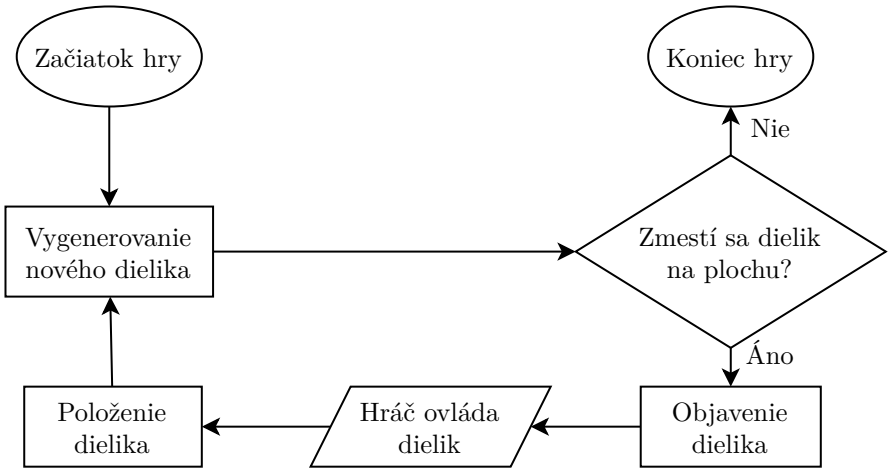
Kapitola 2

Blockstacker hry

Prvá blockstacker hra, totiž originálny Tetris, bola vydaná v roku 1988 a najnovšia oficiálna blockstacker hra, Tetris Effect: Connected, bola vydaná v roku 2020. Medzi týmito hrami celý žáner podstúpil veľké množstvo zmien a vytvorilo sa niekoľko štandardov. V tejto kapitole bude vysvetlený základný princíp, rozobrané používané rozširujúce mechaniky a spôsoby počítanie skóre.

Na konci kapitoly budú takisto uvedené iné blockstacker hry, ktorými je táto práca inšpirovaná.

2.1 Základný princíp



Obr. 2.1: Diagram znázorňuje základný priebeh blockstacker hry.

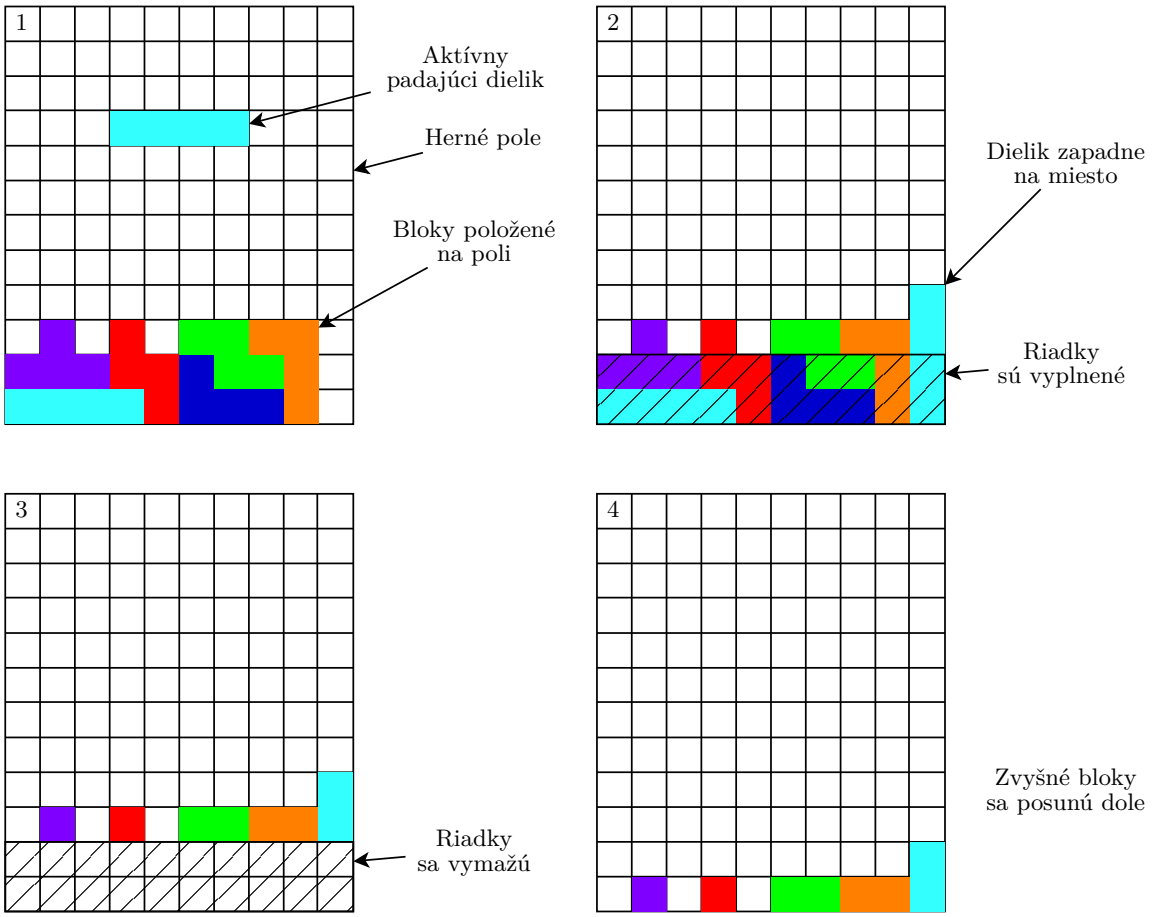
Základným princípom blockstacker hier je pokladanie náhodne generovaných padajúcich dielikov tak, aby sa vyplnili riadky herného poľa. Jeden aktívny dielik padá konštantnou rýchlosťou a keď sa pokúsi padnúť na miesto, ktoré je buď mimo herného poľa alebo zabrané inými položenými dielikmi, tak sa položí.

Hráč má v najjednoduchšej forme hry k dispozícii tri tlačidlá: tlačidlo pre pohyb doprava, tlačidlo pre pohyb doľava a tlačidlo pre rotáciu. Kým dielik padá, hráč ho môže posúvať a otáčať, aby čo najlepšie zapadol medzi ostatné predtým položené dieliky.

V rámci položenia dielika hra prejde všetky riadky herného poľa a zistí, či niektoré z nich neboli po položení dielika vyplnené – ak áno, tak tieto riadky zmaže a bloky v riadkoch nad nimi posunie nižšie.

Po položení dielika sa vygeneruje ďalší dielik. Ak je pre tento dielik miesto na hernej ploche, tak sa objaví a stane sa ďalším aktívnym dielikom. Ak však na hernej ploche nie je miesto pre objavenie ďalšieho dielika, hra končí.

Diagram priebehu hry je uvedený na obrázku 2.1. Priebeh položenia jedného dielika je uvedený na obrázku 2.2.



Obr. 2.2: Obrázok znázorňuje priebeh položenia jedného dielika a následného vymazania riadkov.

V základnom princípe hry je teda cieľom jednoducho prežiť čo najdlhšie pomocou efektívneho vypĺňania riadkov. Jednotlivé implementácie blockstacker hier ďalej pridávajú ďalšie ciele do hry, za ktoré sa dajú získavať body, ktoré nahradia prežitie ako hlavný cieľ hry.

Dostupné dieliky

V typickej blockstacker hre sa zvyčajne používajú iba dieliky zložené zo štyroch blokov. Štandardný termín pre dielik zložený zo štyroch blokov je „tetromino“, ale pre jednoduchosť sa v tejto práci bude ďalej používať slovo dielik. Každý takýto dielik má svoje štandardné meno a farbu. Tieto štandardné mená a farby sa používajú v skoro každej blockstacker hre, keďže umožňujú hráčom jednoducho prechádzať medzi hrami a stále vedieť jednodu-

cho a efektívne spoznávať rôzne dieliky. Štandardné farby a názvy dielikov sú uvedené v tabuľke 2.1. Tieto údaje boli prebrané z Tetris Wiki, zo stránky Tetromino [1].

Dielik	Názov	Štandardná farba
	I dielik	Tyrkysová
	J dielik	Modrá
	L dielik	Oranžová
	O dielik	Žltá
	S dielik	Zelená
	T dielik	Fialová
	Z dielik	Červená

Tabuľka 2.1: Názvy a farby jednotlivých dielikov, štandardne používaných v blockstacker hrách.

Gravitácia

V rámci blockstacker hier sa slovom „gravitácia“ označuje rýchlosť padania dielikov. Takto bude tento výraz použitý aj ďalej v tejto práci, napriek tomu, že to neodpovedá zvyčajnej definícii slova gravitácia. Gravitácia sa udáva v blokoch za snímok. Jeden snímok je štandardne definovaný ako jedna šesťdesiatina sekundy, kvôli typickej vykresľovacej rýchlosti starších hier.

2.2 Dodatočné mechaniky

Základný princíp blockstacker hry je extrémne jednoduchý – cieľom je jednoducho rýchlo a efektívne pokladať dieliky tak, aby v stavbe bolo čo najmenej nevyplnených miest.

Preto boli vymyslené desiatky rôznych mechaník, ktoré hry robia zaujímavejšími, jednoduchšími na hranie a náročnejšími na pochopenie. V tejto sekcii bude popísaná väčšina typických mechaník, ktoré sa v blockstacker hrách zvyknú vyskytovať. Všetky tu popísané mechaniky sú takisto súčasťou tejto bakalárskej práce.

Mechaniky v tejto sekcii budú uvedené podľa toho, ako často sa zvyknú v hrách vyskytovať, od najčastejších po menej časté. Všetky informácie v tejto sekcii sú buď jednoducho odpozorované hraním blockstacker hier, alebo prebrané z Tetris Wiki [1].

2.2.1 Soft drop

Jednou z prvých prídavných mechaník je takzvané „soft drop“ tlačidlo. Stlačením tohoto tlačidla sa gravitácia dočasne zrýchli, čo spôsobí rýchlejšie padanie aktívneho dielika.

V starších hrách bol faktor zrýchlenia gravitácie vždy rovnaký, avšak v novších fanúšikovských hrách je často možné tento faktor zmeniť. Toto nastavenie sa dá nájsť pod názvom SDF – Soft Drop Factor.

2.2.2 Piece spawn delay

Táto mechanika spôsobuje oneskorenie objavenia nasledujúceho dielika po položení. V niektorých hrách je toto oneskorenie aktivované po každom položení dielika – „piece placement delay“, ale najčastejšie sa vyskytuje oneskorenie po vyčistení riadkov – „line clear delay“.

Toto oneskorenie zároveň pridáva hráčom viac času na rozmýšľanie, kam nasledujúci dielik chcú položiť. Okrem toho pri hrách, ktoré vyžadujú čo najväčšiu rýchlosť, pridáva ďalšiu úroveň zložitosti, pretože hráč sa v takom prípade potrebuje snažiť vykonať čo najmenej vyčistení – tým pádom čistiť čo najviac riadkov pomocou štvor-riadkových vyčistení.

2.2.3 Obojstranné rotácie

Pre jednoduchosť ovládania sa často k tlačidlu pre rotáciu pridáva aj druhé tlačidlo, ktoré slúži k otočeniu dielika do druhej strany.

2.2.4 Skóre

Pre zvýšenie komplexnosti blockstacker hier sa už v najskorších hrách miesto času na prežitie zvykla objavovať mechanika skórovania.

V prvotných hrách sa zvykol hodnotiť len počet zmazaných riadkov. Čím viac riadkov teda hráč dokázal počas jednej hry zmazať, tým lepšie bol ohodnotený.

Veľmi rýchlo sa však prišlo na hodnotenie pomocou počtu naraz zmazaných riadkov, pričom jeden naraz zmazaný riadok bol ohodnotený najmenším počtom bodov a štyri naraz zmazané riadky boli hodnotené najvyšším počtom bodov.

Úkon zmazania štyroch riadkov položením jedného dielika zároveň dostal meno – „Tetris“. Slovom Tetris sa teda označujú nielen oficiálne blockstacker hry, ale aj tento úkon.

Počas rokov boli vyvinuté rôzne ďalšie spôsoby skórovania, ktoré budú uvedené ďalej v tejto práci.

2.2.5 Rôzne úrovne hry

Z dôvodu vylepšenia krivky hernej náročnosti bola do blockstacker hier pridaná mechanika postupne sa zvyšujúcej náročnosti – koncept úrovní.

V základnom princípe hry sa obtiažnosť zvyšuje len pokladaním dielikov vyššie a vyššie. Čím vyššie sú dieliky položené, tým rýchlejšie aktívny dielik dosiahne najnižšej možnej polohy a tým kratší čas má hráč na rozhodovanie.

So zvyšujúcou sa úrovňou sa však postupne zvyšuje gravitácia – rýchlosť padania dielikov. Čím rýchlejšia gravitácia je, tým menej času má hráč na rozhodovanie. V posledných úrovniach hier sa gravitácia zvyšuje na absurdné rýchlosti, čím robí efektívne hranie takmer nemožným.

Úroveň sa vo veľkej väčšine hier zvyšuje s počtom vymazaných riadkov. Takisto je zvykom umožniť hráčom vybrať si začínajúcu úroveň, ale do prvého prechodu dať hráčovi viac riadkov, aby sa to v rámci skórovania oplatilo.

V mnohých hrách rôzne úrovne takisto vplývajú na skórovanie. Pri vymazaní riadkov na vyšších úrovniach sa dostáva väčší počet bodov.

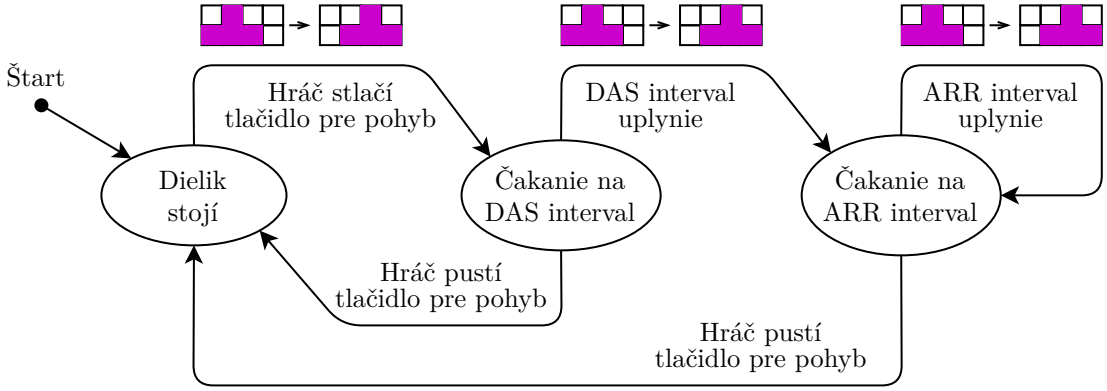
2.2.6 DAS

DAS je skratka pre výraz „Delayed Auto Shift“ – oneskorený automatický posun. Táto mechanika značne zjednodušuje pohybovanie kúskov tým, že ak hráč drží tlačidlo pre pohyb

určitú dobu, tak sa po tej dobe začne kúsok pohybovať sám, až kým hráč dané tlačidlo nepustí alebo nestlačí opačné tlačidlo pre pohyb.

Interval medzi prvotným stlačením tlačidla pre pohyb a začiatkom automatického pohybu je označovaný ako „DAS interval“. Interval medzi jednotlivými automatickými pohybmi je označovaný ako „ARR interval“ – Automatic Repeat Rate. Je treba poznamenať, že napriek názvu Automatic Repeat Rate (rýchlosť automatického opakovania) toto neoznačuje rýchlosť, ale dĺžku intervalu.

Základný priebeh mechaniky DAS je znázornený na obrázku 2.3.



Obr. 2.3: Diagram znázorňuje priebeh mechaniky DAS. Pokiaľ tlačidlo pre pohyb nie je stlačené, dielik stojí. Až hráč tlačidlo stlačí, dielik sa pohne o jeden blok a začne čakať na uplynutie DAS intervalu. Po uplynutí DAS intervalu sa dielik znova pohne o jeden blok a začne čakať na ARR interval. Pri každom nasledovnom uplynutí ARR intervalu sa dielik pohne o ďalší blok, až kým nenarazí na stenu, alebo kým hráč tlačidlo pre pohyb nepustí.

2.2.7 Ukážky

Pre zjednodušenie plánovania bola do blockstacker hier pridaná mechanika ukážok ďalších dielikov. Staršie hry ako Tetris pre NES zvykli ukazovať len jednu ukážku najbližšieho ďalšieho dielika, v novších hrách sa však ukazujú ukážky až na päť alebo šesť dielikov dopredu.

2.2.8 Náhodné generátory

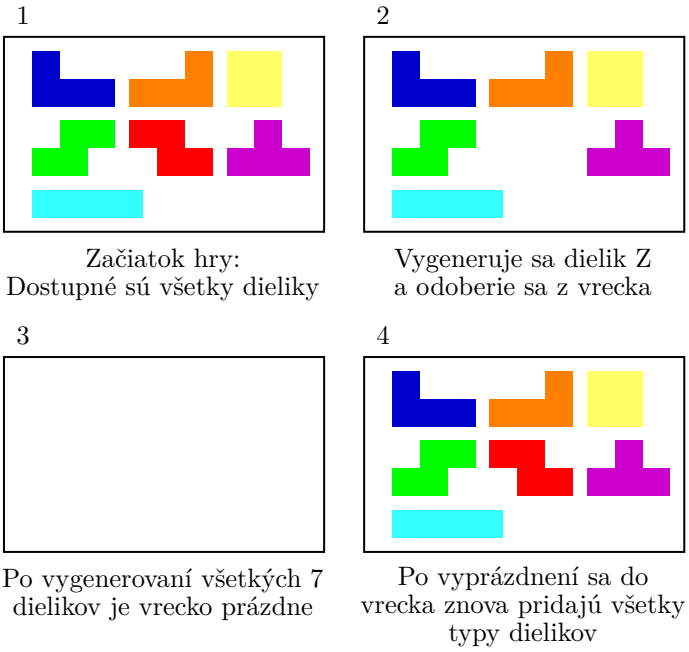
Keďže pri úplne pseudonáhodnom výbere dielikov dokážu blockstacker hry byť pomerne náročné, miesto toho bolo vymyslených niekoľko druhov rôznych generátorov, ktoré do určitej miery obmedzovali náhodnosť vybraných dielikov. Na príklad:

- Jednoduché obmedzovanie opakovania – po vygenerovaní kúska si generátor pamätá, aký kúsok bol naposledy vygenerovaný a ak sa pokúsi vygenerovať rovnaký kúsok hneď po ňom, tak sa náhodné generovanie opakuje. Ak druhý vygenerovaný dielik bude stále rovnaký ako predchádzajúci, tento sa už použije bez ďalších pokusov.
- Pokročilejšie obmedzenie opakovania – v rámci série hier Tetris the Grand Master[1] je používané obmedzenie opakovania, pri ktorom si generátor pamätá nie jeden, ale štyri posledné vygenerované typy dielikov. Pri generovaní použije niekoľko pokusov.

[1] https://www.nintendo.com/store/products/arcade-archives-tetris-the-grand-master-switch/

Ak v rámci pokusov nájde dielik, ktorý nepatrí k štyrom predchádzajúcim, použije ho. Ak nie, tak použije dielik vygenerovaný posledným pokusom.

- Generátor 7-bag – v rámci inicializácie hry sa do „vrecka“ pridá všetkých siedmich možných dielikov. Následne sa z vrecka „ťahá“ po jednom dieliku, čím sa obmedzí nasledujúci výber. Keď vo vrecku nezostane žiadny ďalší dielik, tak sa znova do neho pridá všetkých sedem. Toto je najčastejší spôsob generovania dielikov v moderných blockstacker hrách. Fungovanie tohoto generátoru je znázornené na obrázku 2.4.



Obr. 2.4: Obrázok znázorňuje náhodné generovanie dielikov pomocou generátora 7-bag. Tento generátor používa „vrecko“ na dieliky, z ktorého postupne dieliky pri generovaní vyťahuje.

Okrem tohoto existuje ešte mnoho rôznych ďalších často používaných náhodných generátorov, ale pre jednoduchosť v tejto sekcii nebudú spomínané.

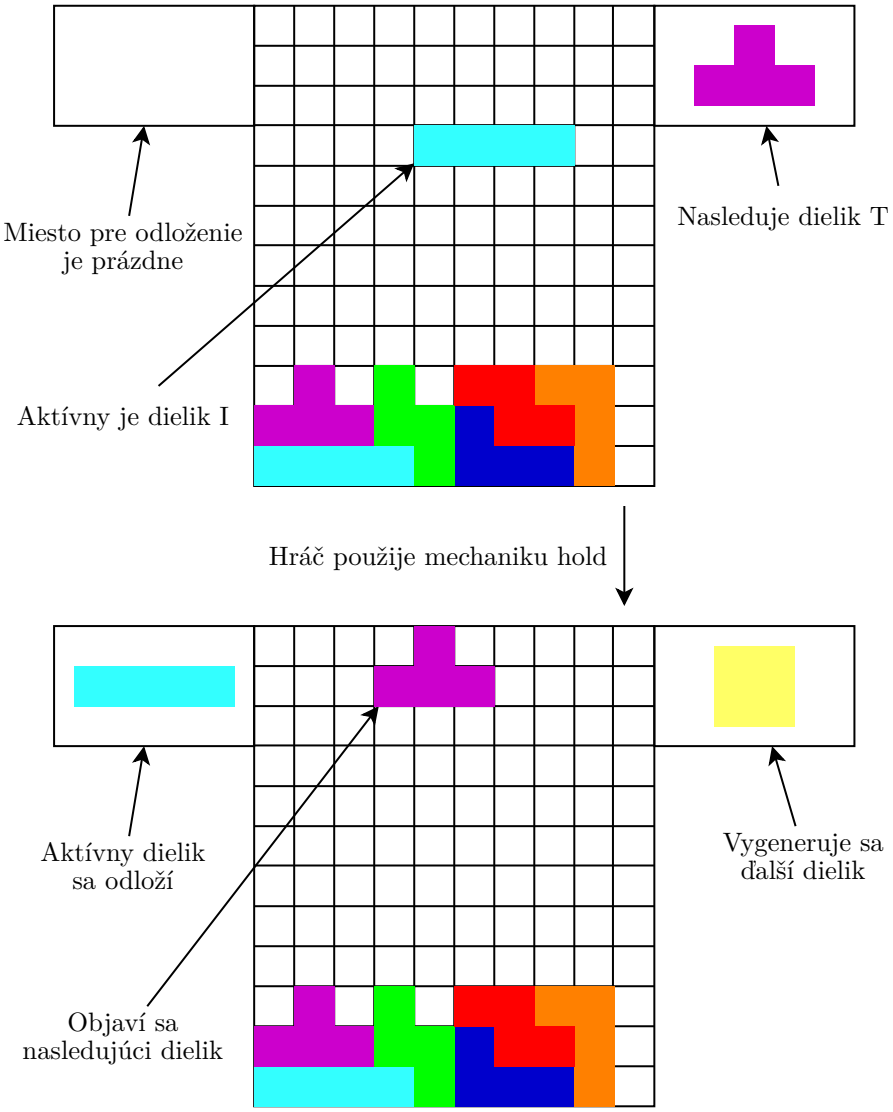
2.2.9 Hold

Pre ďalšie zmenšenie náhodnosti použitých kúskov bola po miernejších generátoroch pridaná takzvaná „hold“ mechanika. Táto mechanika pridáva hold tlačidlo, ktorého stlačením si hráč môže odložiť dielik na neskôr a následne ho vymeniť za iný dielik.

Keď je hold tlačidlo stlačené prvý raz v hre, tak je miesto na odloženie prázdne. Prvý odložený dielik sa jednoducho odloží na miesto a miesto neho hráč dostane nasledujúci dielik. Pri každom nasledovnom stlačení hold tlačidla sa predchádzajúci odložený dielik vymení s aktívnym dielikom.

Po vymenení dieliku pomocou tejto mechaniky sa vždy mechanika zablokuje, až kým hráč dielik nepoloží. Toto zabraňuje hráčovi používať mechaniku donekonečna a získať tým neobmedzený čas na rozmýšľanie o položení dielikov. V rámci tréningu niektoré hry však podporujú aj nastavenie pre nekonečný hold.

Príklad použitia tejto mechaniky je uvedený na obrázku 2.5.



Obr. 2.5: Obrázok znázorňuje použitie mechaniky hold v stave, kedy ešte od začiatku hry nebola použitá.

2.2.10 Hard drop

Pre zrýchlenie hier bolo okrem soft drop tlačidla pridané aj „hard drop“ tlačidlo. Stlačením tohoto tlačidla sa dielik okamžite posunie na najnižšie možné miesto a položí sa.

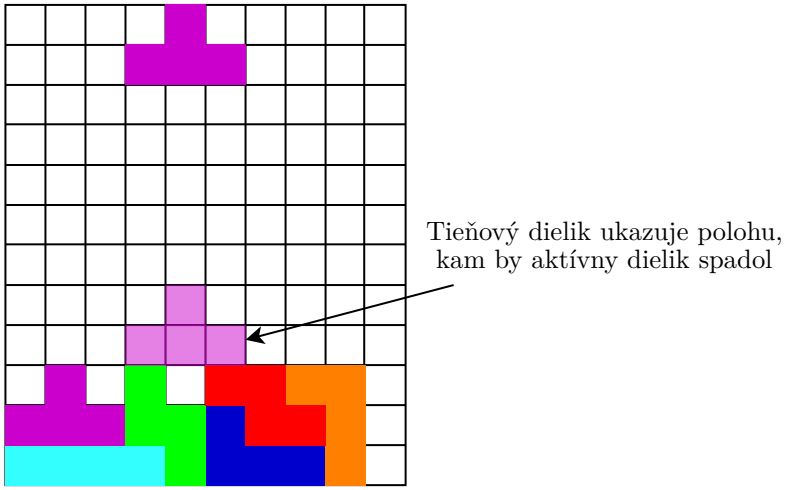
2.2.11 Tieňový dielik

Pre zjednodušenie viditeľnosti, kam dielik dopadne, bola pridaná mechanika tieňového dielika. Tieňový dielik sa zvykne anglicky nazývať „Shadow Piece“ alebo „Ghost Piece“.

Tento dielik nie je skutočnou súčasťou hernej plochy a iba vizuálne ukazuje, kam by aktívny dielik dopadol, ak by sa s ním nič ďalej nedialo. Ak hráč použije hard drop, tak dielik okamžite dopadne na miesto tieňového dielika.

Tieňový dielik sa v hrách takisto zvykne farbiť rovnakými farbami ako aktívny dielik, pre ďalšie zjednodušenie identifikovania aktívneho dielika.

Funkcionalita tieňového dielika je znázornená na obrázku 2.6.



Obr. 2.6: Obrázok znázorňuje výzor typického tieňového dielika. V obrázku je aktívnym dielikom dielik T na vrchu hernej plochy a pod ním je pomocou tieňového dielika zobrazené miesto, kam by spadol.

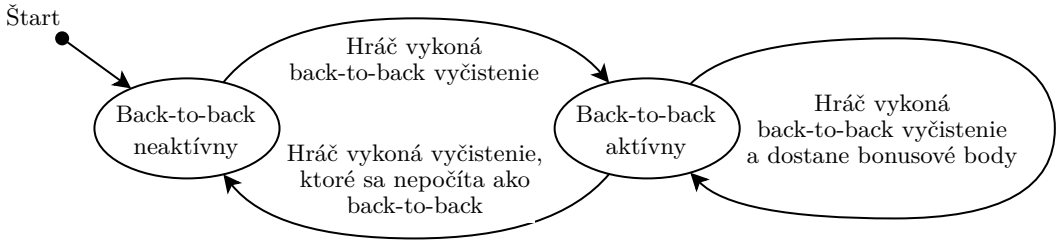
2.2.12 Back-to-back

Back-to-back je skórovacia mechanika, čo odmieňa hráčov, ktorí robia viac „back-to-back vyčistení“ po sebe. Koncept back-to-back vyčistení je rôzny podľa typu hry. Pôvodne boli ako back-to-back identifikované iba štvor-riadkové vyčistenia – Tetrisy.

Ak hráč urobí back-to-back vyčistenie prvý raz, je to základ pre začatie back-to-back reťazca. Ak hráč spraví hneď po back-to-back vyčistení druhé back-to-back vyčistenie, tak toto druhé vyčistenie dostáva bodové ocenenie. V niektorých hrách sa zároveň počíta, koľko back-to-back vyčistení bolo v reťazci urobených a za dlhšie reťazce sa dostáva viac bodov.

Back-to-back reťazec je prerušený vyčistením riadkov spôsobom, ktorý nie je počítaný ako back-to-back vyčistenie. Ak hráč položí kúsok a nevyčistí žiadne riadky, tak reťazec zostáva neprerušený.

Priebeh tejto mechaniky je znázornený na obrázku 2.7.



Obr. 2.7: Diagram znázorňuje ako prebieha mechanika back-to-back.

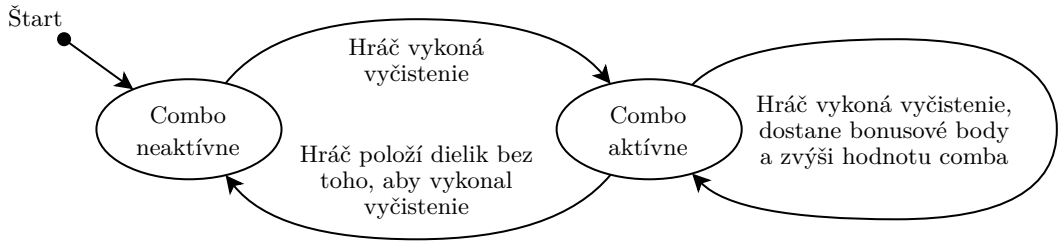
2.2.13 Combo

Combo je skórovacia mechanika, čo odmieňa hráčov, ktorí robia viac vyčistení tesne po sebe. Ak hráč vyčistí dielikom dva riadky a hneď nasledujúcim dielikom vyčistí znova aspoň jeden

riadok, tak dosiahol combo 1. Ak tretím hneď nasledujúcim dielikom vyčistí aspoň jeden riadok, dosiahol combo 2. Čím väčšie combo hráč dosiahne, tým väčšie bodové ocenenie získa.

Combo je prerušené položením dielika, ktorý nevyčistí žiadne riadky.

Priebeh tejto mechaniky je znázornený na obrázku 2.8.



Obr. 2.8: Diagram znázorňuje ako prebieha mechanika combo.

2.2.14 Lock delay

Lock delay – oneskorenie pred uzamknutím je mechanika, ktorá umožňuje hráčom obmedzené ovládanie aktívneho dielika aj pri veľmi vysokej gravitácii.

Funguje tak, že keď sa dielik dotkne zeme, nepoloží sa okamžite, ale čaká sa určitú dobu, označovanú ako lock delay. Táto doba sa dá prerušiť pohnutím alebo otočením aktívneho dielika. Po takomto pohnutí alebo otočení sa znova doba počíta od začiatku, takže hráč môže teoreticky hýbať kúskom na spodku hernej plochy donekonečna.

2.2.15 Hard lock

Hard lock je mechanika na uzamknutie kúsku, ktorá má zabrániť hráčovi zneužívať lock delay a držať jeden dielik donekonečna.

Keďže lock delay sa dá prerušiť pohybom alebo rotáciou, hard lock sa tým prerušiť nedá a vždy začína okamžite potom ako sa dielik dotkne zeme. V oficiálnych hrách sa často na hard lock používa počet pohybov alebo rotácií. V niekoľkých fanúšikovských hrách sa miesto toho používa počet stlačení pohybových či rotačných tlačidiel alebo čas odvtedy, čo sa dielik dotkol zeme.

Táto mechanika sa dá prerušiť jedine tým, že sa dielik posunie na nové najnižšie miesto.

2.2.16 Iné mechaniky pre prehranie hry

V základnom princípe hry hráč zvykne prehrať vtedy, keď nie je miesto na objavenie sa najbližšieho dielika. Niektoré hry však predstavili iné spôsoby ako prehrať hru.

Štandardný názov pre prehru blockstacker hry je „top out“. Top out môže nastať kvôli jednému z nasledujúcich dôvodov:

1. „Block out“ – block out nastáva, ak nie je miesto na objavenie sa najbližšieho dielika. Toto je štandardný spôsob prehry.

2. „Lock out“ – lock out nastáva, ak je dielik položený nad určitou hranicou. Pre lock out musia byť všetky bloky dielika položené nad touto hranicou.

3. Čiastočný lock out – čiastočný lock out nastáva, ak je aspoň jeden blok dielika položený nad lock out hranicou.

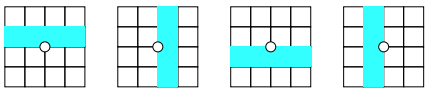
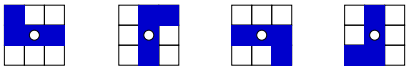
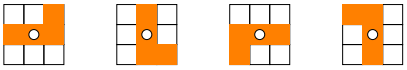
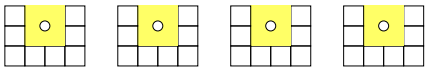
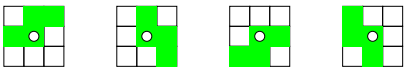
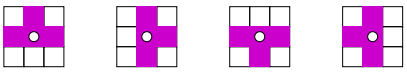
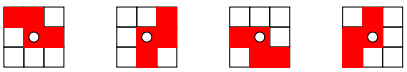
Existuje ešte jeden spôsob prehrania blockstacker hry, ktorý sa nazýva „garbage out“. Toto väčšinou nastáva iba v hrách pre viacerých hráčov, preto je to mimo rozsahu tejto práce.

2.2.17 Rotačné systémy

Ak sa hráč v základnom princípe hry pokúsi rotovať dielik do polohy, v ktorej by nepasoval do herného poľa, tak sa rotácia jednoducho zruší. Rotačné systémy sú mechanika, ktorá pridáva takzvané „kicky“ – kopnutia.

Ak sa rotácia nemôže vykonať, tak sa hra pokúsi „odkopnúť“ dielik do inej polohy. Takýchto skúšok sa vykoná toľko, koľko je v rotačnom systéme kickov pre daný dielik pri danej rotácii.

Najznámejším takýmto rotačným systémom je SRS – Super Rotation System alebo Standard Rotation System. Informácie o tomto rotačnom systéme sú prebrané z Tetris Wiki, z článku Super Rotation System [1]. Tento systém používa štyri kicky pre každú rotáciu. Kicky pre všetky dieliky sú rovnaké, s výnimkou I dielika, ktorý má vlastnú tabuľku kickov. Základné rotácie dieliov v rotačnom systéme SRS sú zobrazené na tabuľke 2.2. Kicky rotačného systému SRS sú zobrazené v tabuľkách 2.3 a 2.3.

Názov dielika	Stav 0	Stav 1	Stav 2	Stav 3
I dielik				
J dielik				
L dielik				
O dielik				
S dielik				
T dielik				
Z dielik				

Tabuľka 2.2: Tabuľka znázorňuje základné rotácie všetkých dielikov v SRS. Pre lepšiu pochopiteľnosť je navyše u každého dieliku bielou bodkou znázornené jeho rotačné centrum. Stav 0 je stav, v ktorom sa dielik objaví, v stave 1 je dielik otočený o 90 stupňov v smere hodinových ručičiek, atď.

Rotácia	Skúška 0	Skúška 1	Skúška 2	Skúška 3	Skúška 4
$0 \rightarrow 3$	(0, 0)	(1, 0)	(1, 1)	(0, -2)	(1, -2)
$0 \rightarrow 1$	(0, 0)	(-1, 0)	(-1, 1)	(0, -2)	(-1, -2)
$1 \rightarrow 0$	(0, 0)	(1, 0)	(1, -1)	(0, 2)	(1, 2)
$1 \rightarrow 2$	(0, 0)	(1, 0)	(1, -1)	(0, 2)	(1, 2)
$2 \rightarrow 1$	(0, 0)	(-1, 0)	(-1, 1)	(0, -2)	(-1, -2)
$2 \rightarrow 3$	(0, 0)	(1, 0)	(1, 1)	(0, -2)	(1, -2)
$3 \rightarrow 2$	(0, 0)	(-1, 0)	(-1, -1)	(0, 2)	(-1, 2)
$3 \rightarrow 0$	(0, 0)	(-1, 0)	(-1, -1)	(0, 2)	(-1, 2)

Tabuľka 2.3: Tabuľka znázorňuje kicky v rotačnom systéme SRS pre dieliky J, L, O, S, T, a Z. Prvá súradnica značí pohyb na osi X a druhá súradnica značí pohyb na osi Y. Osa X rastie smerom doprava a osa Y rastie smerom dohora.

Rotácia	Skúška 0	Skúška 1	Skúška 2	Skúška 3	Skúška 4
$0 \rightarrow 3$	(0, 0)	(-1, 0)	(2, 0)	(-1, 2)	(2, -1)
$0 \rightarrow 1$	(0, 0)	(-2, 0)	(1, 0)	(-2, -1)	(1, 2)
$1 \rightarrow 0$	(0, 0)	(2, 0)	(-1, 0)	(2, 1)	(-1, -2)
$1 \rightarrow 2$	(0, 0)	(-1, 0)	(2, 0)	(-1, 2)	(2, -1)
$2 \rightarrow 1$	(0, 0)	(1, 0)	(-2, 0)	(1, -2)	(-2, 1)
$2 \rightarrow 3$	(0, 0)	(2, 0)	(-1, 0)	(2, 1)	(-1, -2)
$3 \rightarrow 2$	(0, 0)	(-2, 0)	(1, 0)	(-2, -1)	(1, 2)
$3 \rightarrow 0$	(0, 0)	(1, 0)	(-2, 0)	(1, -2)	(-2, 1)

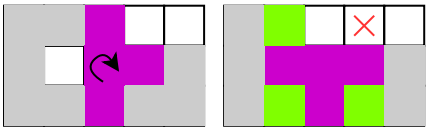
Tabuľka 2.4: Tabuľka znázorňuje kicky v rotačnom systéme SRS pre dielik I. Prvá súradnica značí pohyb na osi X a druhá súradnica značí pohyb na osi Y. Osa X rastie smerom doprava a osa Y rastie smerom dohora.

2.2.18 Spin

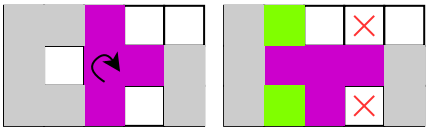
Spin je špeciálny spôsob položenia dieliku, kde hráč tesne pred položením dielik zarotuje do určitej pozície. Toto sa väčšinou rozpoznáva iba u T dielikov. Takéto spiny sa často hodnotia väčším množstvom bodov. U hier, ktoré podporujú mechaniku back-to-back (vysvetlené v sekcii 2.2.12), sa vyčistenia urobené pomocou spinov berú ako „back-to-back vyčistenia“, čiže back-to-back body sa pripočítavajú aj k nim.

T-Spiny sú detekované pomerne zložito. Základné pravidlo je pravidlo rotácie – posledná akcia, čo sa s dielikom stane pred položením, musí byť rotácia.

Potom pravidlo troch rohov – minimálne tri rohy 3x3 slotu, do ktorého sa T dielik položí, musia byť zabrané nejakými inými blokmi. Toto pravidlo je demonštrované na obrázku 2.9.



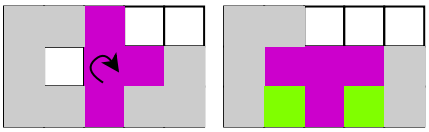
Počíta sa ako T-spin



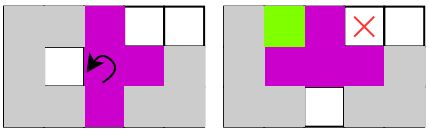
Nepočíta sa ako T-spin

Obr. 2.9: Obrázok znázorňuje pravidlo troch rohov. Dodržanie tohoto pravidla je podmienkou pre identifikovanie položenia dielika T ako spinu.

Na plné T-Spiny zároveň existuje pravidlo dvoch rohov – dva rohy v smere, kam je T dielik otočený, musia byť zabrané. Ak tieto dva rohy zabrané nie sú, tak sa položenie dieliku berie ako tzv. T-Spin mini. Toto pravidlo je demonštrované na obrázku 2.10.



Plný T-spin

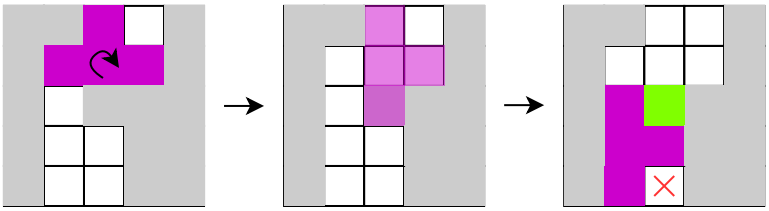


T-spin mini

Obr. 2.10: Obrázok znázorňuje pravidlo dvoch rohov. Dodržanie tohoto pravidla je podmienka pre identifikovanie položenia dielika T ako plného spinu.

Na pravidlo dvoch rohov avšak existuje výnimka – v určitých prípadoch sa berie spin ako plný aj ak je zaplnený iba jeden z rohov, kam je T dielik otočený. Tieto prípady nastávajú vtedy, keď sa použijú špecifické kicky z rotačného systému. V systéme SRS,

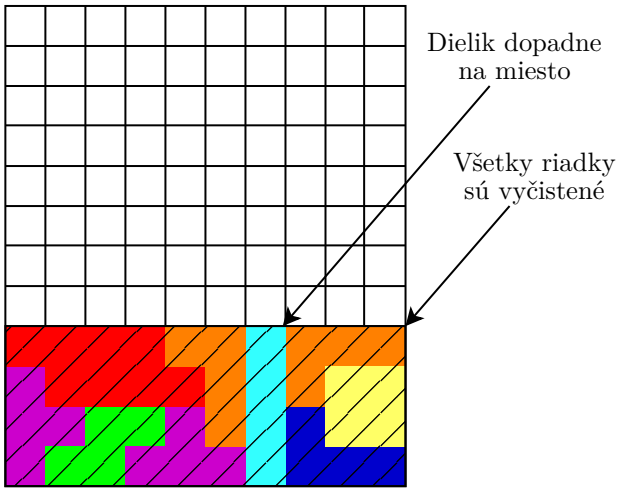
ktorý je najčastejšie používaným rotačným systémom, sú to kicky (1, -2) a (-1, -2). Toto sa dá využiť na robenie plných T-Spinov aj za okolností, kedy by boli brané len ako T-Spiny mini. Využitie takejto výnimky je znázornené na obrázku 2.11.



Obr. 2.11: Obrázok znázorňuje výnimku na pravidlo 2 rohov. V tomto prípade hráč zarotuje dielik a použije sa kick (-1, -2), ktorý dostane dielik na miesto. Vďaka použitiu tohoto kicku sa toto počíta ako plný T-Spin, nie ako T-Spin mini.

2.2.19 All clear

All clear alebo perfect clear nastáva, keď hráč vyčistí všetky bloky na hracej ploche. Toto býva často oceňované veľkým množstvom bodov. Príklad vykonania all clearu je dostupný na obrázku 2.12.



Obr. 2.12: Obrázok znázorňuje položenie dielika takým spôsobom, že nastane all clear.

2.2.20 Rotácia o 180 stupňov

Niektoré fanúšikovské hry mimo obojstranných rotácií pridávajú ešte jedno tlačidlo pre rotáciu – rotáciu o 180 stupňov. Toto znova zjednodušuje hráčom ovládanie a zmenšuje počet stlačení potrebných na položenie dieliku.

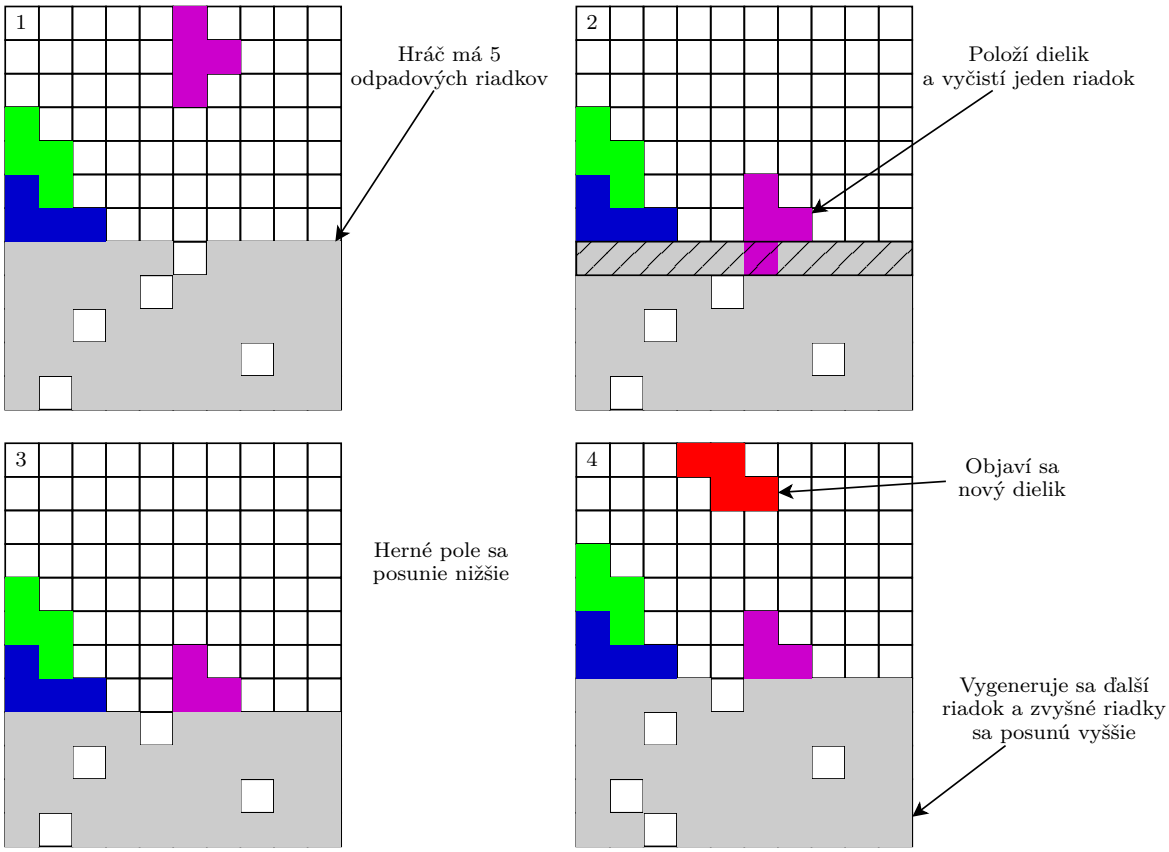
2.2.21 Generovanie odpadových riadkov

Niektoré hry podporujú mechaniku generovania takzvaných „odpadových riadkov“, anglicky nazývaných garbage lines alebo cheese lines kvôli podobnosti syru s dierami. Takéto riadky sa zvyknú objavovať na spodnej časti hernej plochy a posúvať všetky riadky nad nimi vyššie.

Táto mechanika je užitočná hlavne pre hry pre viac hráčov, kde je často používaná ako spôsob útočenia na protivníkov. Vykonaním určitého typu vyčistenia sa protivníkom posielajú odpadové riadky, ktoré jeho hernú plochu dostávajú vyššie a tým pádom potenciálne bližšie k prehre.

V hrách pre jedného hráča je táto mechanika často používaná na tréning vyčisťovania náročne ukladaných blokov. Takisto v určitých herných módoch môže byť využitá na dostanie hernej plochy do stavu, ktorý vyžaduje od hráča, aby položil dieliky určitým spôsobom.

Príklad generovania odpadových riadkov je zobrazený na obrázku 2.13.



Obr. 2.13: Obrázok znázorňuje, ako vyzerajú odpadové riadky a ako prebieha ich generovanie.

V tejto sekcii je uvedený popis moderných blockstacker hier, ktorými bola táto bakalárska práca inšpirovaná.

2.3 Iné moderné blockstacker hry

V tejto sekcii budú uvedené hry, ktoré boli najväčšou inšpiráciou pre túto bakalársku prácu.

Oficiálne Tetris hry reprezentuje Tetris Effect, graficky pokročilá hra pre virtuálnu realitu s veľkým množstvom zaujímavých a originálnych herných módov.

Staršia generácia hier je reprezentovaná hrou Jstris, vyvíjanou českým vývojárom, ktorá je od roku 2014 až dodnes stále obľubovaná hráčmi pre responzívne ovládanie, detailné rebríčky a jednoducho prístupný mód pre viac hráčov. Takisto je jej veľkou výhodou nízka náročnosť, vďaka ktorej je možné Jstris hrať skoro na akomkoľvek zariadení.

Nakoniec je spomenutá veľmi populárna hra TETR.IO, ktorá rýchlo získala veľkú popularitu vďaka mnohým možnostiam, výbornému dizajnu grafického rozhrania a kompetitívnemu módu „Tetra League“.

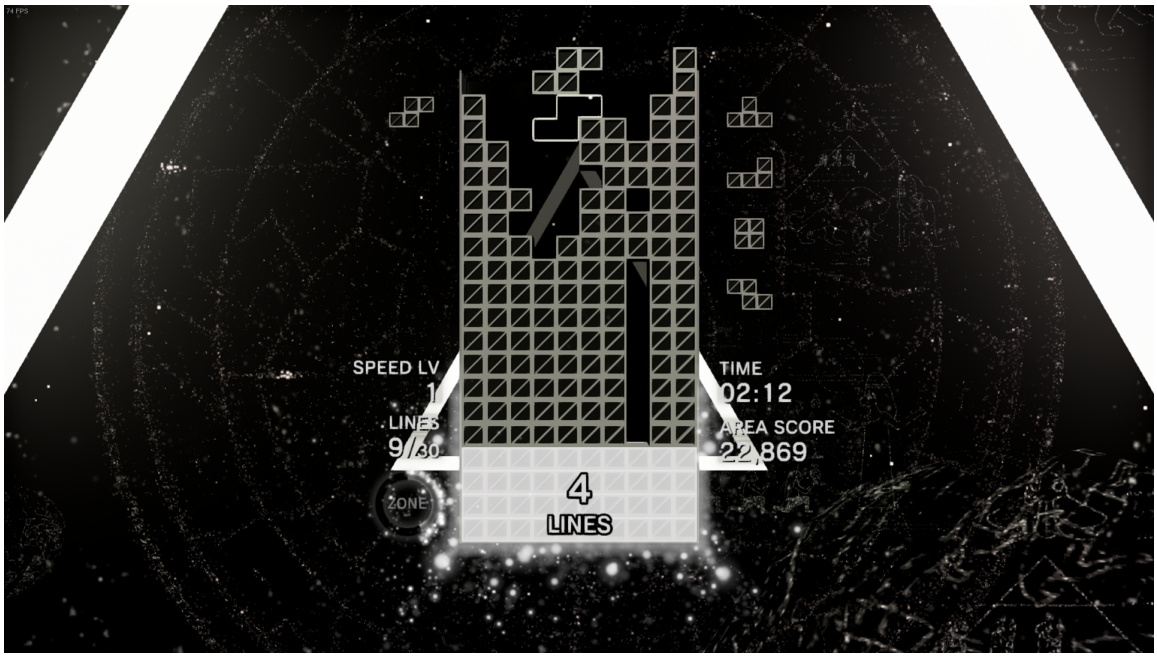
2.3.1 Tetris Effect

Tetris Effect je jednou z najnovších oficiálnych blockstacker hier. Táto hra podporuje veľké množstvo rôznych herných módov, je jedinou blockstacker hrou určenou pre hranie vo virtuálnej realite a takisto je veľmi známa tým, ako prispôsobuje hudbu hry tomu, čo hráč robí, miesto používania jednoduchých zvukových efektov.

Hlavnou časťou hry je takzvaný „Journey mód“, v ktorom hráč prechádza medzi rôznymi vizuálne a hudobne odlišnými časťami. V každej časti je cieľom pre hráča vyčistiť určitý počet riadkov. Podľa toho, akým spôsobom hráč riadky čistí, sa hráčovi takisto udeľuje skóre. Na konci každej časti je hráč ohodnotený známkou od F (prehra) do SS (vynikajúce skóre).

V rámci Journey módu bola taktiež implementovaná špeciálna mechanika s názvom Zone. Táto mechanika funguje na princípe zastavenia herného času. Hráč si dokáže nabiť zdroj na používanie Zone čistením riadkov a keď použije tlačidlo pre aktiváciu Zone, tak sa tento zdroj začne odpočítavať. Kým zdroj nevyprchá, v hre sa pozastaví gravitácia a všetky vyčistené riadky sa budú miesto miznutia presúvať na spodok hernej plochy. Týmto spôsobom môže hráč teoreticky „naraz“ vyčistiť až 23 riadkov.

Výzor hry Tetris Effect počas hrania je zobrazený na obrázku 2.14.



Obr. 2.14: Obrázok ukazuje Tetris Effect v hre.

Okrem Journey módu hra obsahuje niekoľko ďalších módov hry pre jedného hráča:

- „Marathon“ – hráč sa snaží dosiahnuť čo najvyššie skóre v rámci 150 vyčistených riadkov.

- „Zone Marathon“ – podobné ako Maratón, avšak v tejto variante je prístupná mechanika Zone.
- „Ultra“ – hráč sa snaží dosiahnuť čo najvyššie skóre v rámci troch minút.
- „Sprint“ – hráč sa snaží čo najrýchlejšie vyčistiť štyridsať riadkov.
- „Master“ – hráč sa snaží dosiahnuť čo najvyššie skóre s neobmedzeným časom, avšak v tomto móde sa vyskytuje instantná gravitácia a lock delay (sekcia 2.2.14) sa postupne znižuje, čo robí hru extrémne rýchlou.
- „Classic Score Attack“ – špeciálny herný mód, ktorý sa pokúša napodobniť klasický Tetris pre NES. Tento mód používa špeciálny levelovací systém a núti hráča používať určité ovládanie, typické pre Tetris pre NES.
- „Chill Marathon“ – podobné ako Maratón, avšak bez systému úrovní a bez limitu 150 riadkov.
- „All Clear“ – hráč sa snaží v rámci časového limitu urobiť čo najviac all clearov (sekcia 2.2.19). Pri začatí hry sa herné pole zmení tak, aby hráčovi umožnilo urobiť all clear iba jedným spôsobom. Po každom úspešnom all cleare sa hráčovi pridá niekoľko sekúnd do časového limitu a herné pole sa znovu zmení. Po neúspešnom all cleare sa herné pole takisto znovu zmení.
- „Combo“ – podobné ako all clear mód, avšak pri tomto móde sa hráč pokúša pri každom stave hernej plochy urobiť čo najdlhší combo reťazec (sekcia 2.2.13).
- „Target“ – podobné ako all clear a Combo módy, ale v tomto prípade sa na plochu umiestňujú špeciálne „terčové“ bloky, ktoré slúžia ako bodovanie pre hráča. Hráč dostáva body len ak vyčistí riadky, ktoré obsahujú tieto terčové bloky. Potom, ako ich vyčistí, sa herné pole zmení do stavu s novými terčovými blokmi.
- „Countdown“ – hráč sa snaží v určitom časovom intervale získať čo najviac bodov. Avšak po položení určitého počtu blokov sa na poli hráča objaví I dielik. Za bloky vyčistené týmto I dielikom hráč dostáva bonusové body. Miesto objavenia tohoto dielika a je na hernom poli jasne naznačené. Niekedy sa týchto dielikov objaví aj viac.
- „Purify“ – hráč sa snaží v rámci časového limitu vyčistiť čo najviac „znečistených“ blokov. Tieto znečistené bloky sa na hernom poli objavujú v určitých intervaloch.
- „Mystery“ – hráč sa snaží vyčistiť 150 riadkov a dosiahnuť čo najväčšie skóre počas toho, čo sa jeho herná plocha v určitých intervaloch neustále mení náhodnými spôsobmi. Medzi efekty, ktoré môžu nastať, patria napríklad: obrátenie hernej plochy naopak, znemožnenie použitia niektorých herných mechaník, pridávanie bodov navyše, generovanie dvojnásobne veľkých dielikov a iné.

Okrem týchto herných módov Tetris Effect podporuje aj hru pre viacerých hráčov a poskytuje svetové rebríčky pre všetky svoje herné módy.

Z predchádzajúcich spomenutých herných mechaník však nepodporuje nastavovanie vlastného DAS a ARR (sekcia 2.2.6), soft drop faktor a ani rotácie dielikov o 180 stupňov.

2.3.2 Jstris

Jstris je online blockstacker hra pre jedného alebo viac hráčov, vyvíjaná od roku 2014. Je zameraná hlavne na mód hry pre viacerých hráčov, ktorý je založený na posielaní „odpadových“ riadkov protihráčom. Hráč, ktorý prežije najdlhšie, vyhráva.

Výzor hry Jstris je zobrazený na obrázku 2.15.

Obr. 2.15: Obrázok ukazuje hru Jstris v móde hry pre viacerých hráčov.

Okrem módu pre viacerých hráčov však podporuje aj rôzne módy pre jedného hráča s globálnymi a štátnymi rebríčkami. Medzi tieto módy patria:

- „Sprint“ – hráč sa snaží vyčistiť určitý počet riadkov čo najrýchlejšie.
- „Cheese race“ – hráč sa snaží vyčistiť určitý počet náhodne generovaných odpadových riadkov s dierami čo najrýchlejšie.
- „Map Downstack“ – hráč sa snaží vyčistiť bloky, ktoré iný hráč nakreslil pomocou nástroja na vytváranie máp pre Jstris. Tento mód sa rôzni podľa toho, akú mapu si hráč na začiatku vyberie.
- „Survival“ – hráč sa snaží prežiť čo najdlhšie, zatiaľ čo sa mu donekonečna v určitých časových intervaloch posielajú odpadové riadky.
- „Ultra“ – hráč sa snaží za čas dvoch minút dosiahnuť čo najlepšie skóre.
- „20TSD“ – hráč sa snaží vyčistiť čo najviac riadkov po sebe pomocou T-Spin vyčistení, ktoré vyčistia dva riadky.
- „PC Mode“ – hráč sa snaží po každých desať dielikoch urobiť all clear (sekcia 2.2.19) tak dlho, ako je to možné.

- „Practice“ – špeciálny mód bez možnosti ukončiť hru a bez rebríčkov.
- „Bots“ – hráč sa podľa štandardných pravidiel pre viacerých hráčov snaží poraziť vybraného robota v hre jeden proti jednomu.

Jstris podporuje všetky dodatočné mechaniky spomenuté v sekcii 2.2. Takisto podporuje možnosť voľne si meniť hodnoty DAS a ARR (sekcia 2.2.6) a vybrať si niekoľko rôznych rýchlostí pre soft drop (sekcia 2.2.1).

Jstris takisto ku všetkým svojim herným módom umožňuje ukladať súbory, vďaka ktorým sa dá hra prehrať od začiatku až do konca. Všetky hry, ktoré sú uložené v rebríčkoch, sa dajú takýmto spôsobom jednoducho prehrať. V prípade hier pre viac hráčov je však možné zobraziť len jedného hráča naraz.

2.3.3 TETR.IO

TETR.IO je momentálne najpopulárnejšou blockstacker hrou a zároveň aj najväčšou inšpiráciou pre túto bakalársku prácu. Podobne ako u hry Jstris je najväčšou súčasťou tejto hry mód pre viacerých hráčov.

Výzor tejto hry pri hre pre viacerých hráčov je zobrazený na obrázku 2.16.

Obr. 2.16: TETR.IO uprostred hry pre viac hráčov

TETR.IO podporuje relatívne malé množstvo herných módov oproti ostatným moderným blockstacker hrám:

- „Tetra League“ – súťažný mód, v ktorom sú hráči proti sebe vyberaní podľa schopností a vždy hrajú jeden proti jednému. Toto je hlavný mód tejto hry, keďže žiadna iná blockstacker hra zadarmo podobný mód natívne neposkytuje,
- „Quick Play“ – mód pre viacerých hráčov podobný Jstrisu,

- „Sprint“ – ekvivalent módu Sprint v Jstrise, avšak v TETR.IO je podporovaný iba cieľ štyridsať riadkov,
- „Ultra“ – ekvivalent módu Ultra v Jstrise s iným spôsobom počítania bodov a s pridaným systémom úrovní.

Okrem týchto herných módov však podporuje pomerne extenzívne možnosti vytvárania herných nastavení, či už pre jedného hráča alebo pre viacerých hráčov. Tieto nastavenia sa dajú uložiť do súboru a preniesť na iný počítač, kde sa z daného súboru dajú načítať.

TETR.IO takisto podporuje všetky herné mechaniky spomenuté v sekcii 2.2. Takisto poskytuje možnosť voľne si meniť hodnoty DAS a ARR (vysvetlené v sekcii 2.2.6) a soft drop faktor (vysvetlený v sekcii 2.2.1). K tomu pridáva ešte niekoľko možností ovládania, ktoré môžu niektorým hráčom zjednodušiť hru.

TETR.IO takisto ku všetkým svojim herným módom umožňuje ukladať súbory, vďaka ktorým sa dá hra prehrať od začiatku až do konca. Pre každého hráča v rebríčkoch ukladá desať najlepších výsledkov v módoch pre jedného hráča a desať posledných hier v móde Tetra League. Toto je v TETR.IO však možné iba pri maximálnom počte dvoch hráčov.

Kapitola 3

Popis funkcionality

Táto kapitola obsahuje popis funkcionality tejto bakalárskej práce a jej jednotlivých častí.

- V sekcii 3.1 bude rozobraná štruktúra aplikácie hry.
- V sekcii 3.2 budú rozobrané všeobecné vlastnosti aplikácie hry.
- Sekcia 3.3 obsahuje popis podporovaných herných mechaník.
- Nakoniec bude v sekcii 3.4 popísané ako je možné hru upraviť aby vyzerala a znela tak, ako si hráč predstavuje.

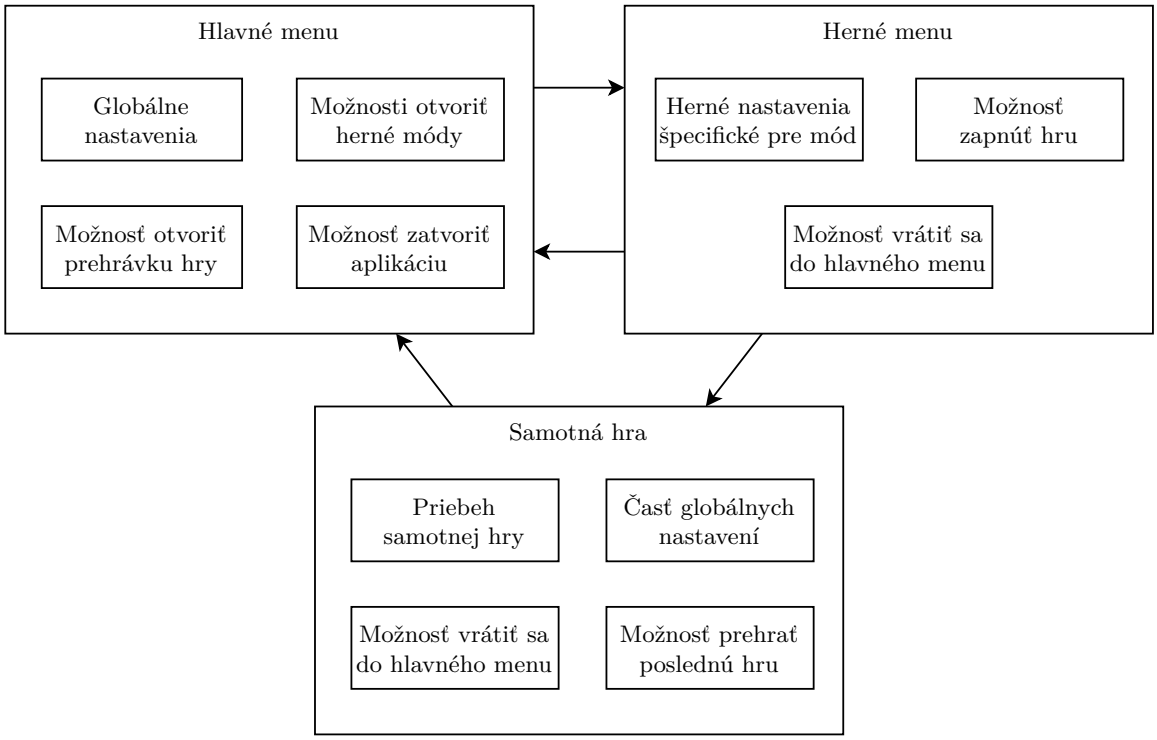
3.1 Štruktúra aplikácie a priebeh hry

Aplikácia hry pozostáva z niekoľkých častí:

1. Hlavné menu s možnosťami vybrania herného módu, úpravy globálnych nastavení a ukončenia hry.
2. Herné menu s možnosťami upravenia nastavení, ktoré sa vzťahujú na hru vybranú v hlavnom menu.
3. Samotná hra.

Hneď po spustení hry sa načítajú perzistentné užívateľské nastavenia, ktoré ovládajú chovanie aplikácie hry a niektoré aspekty jednotlivých hier. Všetky herné komponenty po tomto kroku vedia, ako sa majú načítať a správať počas hry.

Štruktúra aplikácie a prechody medzi jej jednotlivými časťami sú zobrazené na obrázku 3.1.



Obr. 3.1: Diagram ukazuje štruktúru aplikácie hry a užívateľské možnosti v jednotlivých častiach aplikácie.

Hlavné menu

Hlavné menu slúži ako primárny rázcestník pre možnosti hráča. Obsahuje zoznam všetkých herných módov, ktoré hra poskytuje, vrátane možnosti vlastného nastavenia hry.

Ďalej obsahuje užívateľské rozhranie k zmene globálnych nastavení, ktoré sú vždy načítané z perzistentného úložiska pri spustení hry. Pri zmene nastavení sú vyvolávané udalosti, na ktoré dokážu časti hry reagovať a okamžite sa meniť aby odpovedali novým nastaveniam. Nastavenia sú počas priebehu hry udržiavané v pamäti a pri ukončení hry sa automaticky uložia.

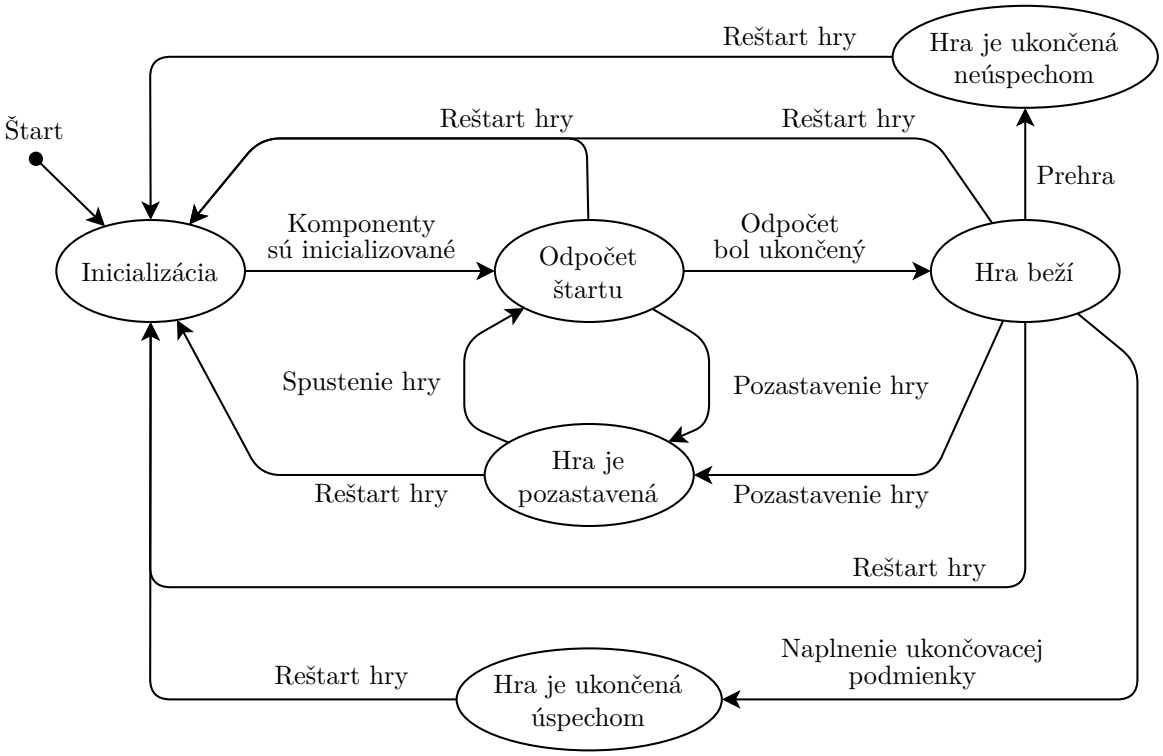
Herné menu

Herné menu obsahuje všetky nastavenia, ktoré sú relevantné k vybranej hre. Pre vstavané herné módy je tu možné meniť nastavenia ktoré sa vzťahujú k zobrazeniu a hudbe hry a takisto začiatočnú úroveň. Pre vlastnú hru je možné tu upravovať všetky nastavenia, ktorá hra poskytuje.

3.1.1 Herná časť aplikácie

Po načítaní samotnej hry z herného menu sa spustí stavový stroj hry. Hra začína v stave inicializácie, kedy prebieha nastavenie všetkých komponentov hry tak, aby vyhovovali herným a globálnym nastaveniam.

Až sa všetky komponenty inicializujú, spustí sa odpočet štartu hry. Počas odpočtu je možné hru pozastaviť alebo reštartovať použitím priradených tlačidiel.



Obr. 3.2: Stavový diagram priebehu hry

Po ukončení odpočtu sa hra spustí a začnú sa registrovať vstupy užívateľa. Počas behu hry je možné hru znovu pozastaviť alebo reštartovať.

Keď je hra pozastavená, dajú sa meniť určité globálne nastavenia, ktoré na priebeh hry nemajú vplyv. Z pozastaveného stavu hry sa dá prejsť buď znovu do inicializácie alebo do odpočtu štartu hry.

Ak hráč splní podmienku pre ukončenie hry, ktorá bola nastavená v herných nastaveniach, tak je hra úspešne ukončená a je možné ju začať odznovu.

Podobne, ak hráč prehrá (nastane top out), tak je hra ukončená neúspechom a hráč sa môže pokúsiť ju reštartovať a hrať odznovu.

Stavový diagram priebehu hry je zobrazený na obrázku 3.2.

Počas hry medzi sebou jednotlivé časti potrebujú komunikovať a reagovať na dané správy. Na túto komunikáciu je v tejto bakalárskej práci použitý takzvaný mediátor, ktorý spravuje všetky udalosti, ktoré počas hry môžu nastať. Pri inicializácii hry si jednotlivé komponenty u mediátora dokážu zaregistrovať funkcie, ktoré dokážu na dané udalosti reagovať. Až udalosť nastane, tak sa podľa priority budú obslužné funkcie postupne spúšťať.

3.2 Vlastnosti aplikácie

Aplikácia okrem hrania samotného blockstacker hier podporuje rôzne ďalšie vlastnosti. V tejto kapitole bude popísaná väčšina menších vlastností, ktoré nie sú priamo späté so samotnou hrou.

3.2.1 Prehrávanie hier odznova

Po úspešnom ukončení jednej hry aplikácia umožňuje automatické ukladanie súborov, ktoré obsahujú všetky potrebné informácie k prehraní hry odznova. Jeden takýto súbor obsahuje všetky herné nastavenia danej hry, zoznam dosiahnutých zaznamenávaných štatistík a zoznam použitých užívateľských akcií. Je zaznamenané každé stlačenie a pustenie tlačidiel, ktoré hru ovládajú.

Okrem tlačidiel sú taktiež zaznamenávané všetky časy položenia dielikov.

Hru je možné prehrať hneď po dokončení, alebo načítaním súboru s potrebnými informáciami. Vďaka zaznamenávaniu časov akcií, ktoré boli v hre použité na ovládanie, a seedu použitého náhodného generátora je prehraná hra identická ako pôvodné užívateľské akcie.

Rozhranie prehrávok hier takisto umožňuje hru prehrať s inými počítadlami štatistík a v rámci prehrávky zmeniť rýchlosť priebehu času, preskočiť na akýkoľvek moment v hre a prípadne aj prehrávku pozastaviť.

3.2.2 Priradenie užívateľských akcií k preferovaným tlačidlám

Pre optimálny užívateľský zážitok je potrebné aby hráč mohol nastaviť, ktoré tlačidlá ovládajú ktoré užívateľské akcie v hre. Táto bakalárska práca podporuje nastavenie tlačidiel pre nasledujúce užívateľské akcie:

- Posun dielika doľava,
- posun dielika doprava,
- soft drop (vysvetlené v sekcii 2.2.1),
- hold (vysvetlené v sekcii 2.2.9),
- hard drop (vysvetlené v sekcii 2.2.10),
- rotácia po smere hodinových ručičiek,
- rotácia proti smeru hodinových ručičiek,
- rotácia o 180 stupňov,
- posun ku ľavej stene,
- posun ku pravej stene,
- reštart hry,
- pozastavenie hry.

Ku každej užívateľskej akcii je možné priradiť tri tlačidlá klávesnice alebo herného ovládača. V prípade, že užívateľ nastaví jedno tlačidlo na dve alebo viac rôznych akcií, hra ho upozorní zmenou farby v užívateľskom rozhraní.

Užívateľovi je takisto umožnené zrušiť východzie priradenia tlačidiel k akciám.

3.2.3 Základné prispôsobenie hry

Hru je možné rôzne prispôsobiť pre optimálny užívateľský zážitok. Medzi jednoduché nastavenia prispôsobenia hry patria:

- Viditeľnosť hernej plochy,
- priblíženie hernej plochy,
- viditeľnosť mriežky hry,
- viditeľnosť tieňového dielika,
- viditeľnosť pozadia hry,

- úprava hlasitosti hudby a zvukových efektov,
- možnosť zobrazenia „varovného dielika“, ktorý ukazuje, kde sa objaví ďalší dielik,
- úprava výšky hernej plochy, pri ktorej sa začne varovný dielik zobrazovať,
- výber schémy pomenovania súborov prehrávok,
- možnosť priblíženia hernej plochy počas hry,
- možnosť zmeny pozície hernej plochy počas hry,
- možnosť výberu sfarbenia tieňového dielika,
- možnosť zobrazenia varovania v prípade, že hra beží a aplikácia nemá focus,
- možnosť automatického pozastavenia hry v prípade, že aplikácia stratí focus,
- možnosť automatického umlčania aplikácie hry v prípade, že aplikácia nemá focus,
- možnosť prehrávania zvukov príslušných k typom najbližších dielikov,
- možnosť zmeniť režim aplikácie na režim fullscreen, režim okna bez okrajov alebo režim obyčajného okna,
- možnosť vybrania preferovanej vykresľovacej rýchlosti hry,
- možnosť zobrazovania hernej aktivity v aplikácii Discord,
- možnosť zmeny umiestnenia perzistentného úložiska herných nastavení a automatické presunutie uložených súborov na nové miesto,
- možnosť exportovania herných nastavení do súboru alebo ich importovania z predtým uloženého súboru.

3.3 Herné mechaniky

V tejto sekcii budú rozpísané všetky podporované herné mechaniky. U štandardných mechaník, ktoré boli spomínané v sekcii 2.2, budú popísané len zmeny a rozšírenia, ktoré boli v rámci tejto bakalárskej práce navrhnuté. Okrem nich táto bakalárska práca podporuje aj ďalšie mechaniky, ktoré sa v iných hrách nevyskytujú.

3.3.1 Štandardné mechaniky

Tieto mechaniky sú v rámci tejto bakalárskej práce implementované, ale neboli zásadne zmenené ani rozšírené. Sú to nasledovné mechaniky:

- Rotácie po smere a proti smeru hodinových ručičiek a o 180 stupňov. Rotácie o 180 stupňov je možné v nastaveniach herného módu zakázať.
- Piece spawn delay (vysvetlené v sekcii 2.2.2). V nastaveniach herného módu je túto mechaniku možné nastaviť pomocou nasledujúcich možností:
    1. Piece placement delay – oneskorenie po položení každého dielika. Ďalší dielik sa v hre objaví až po uplynutí tohoto intervalu.

2. Line clear delay – oneskorenie po vyčistení riadkov. V prípade, že boli posledným položením dielika vyčistené nejaké riadky, sa ďalší dielik objaví až po uplynutí tohoto intervalu.

Celkový interval piece spawn delay sa teda v hre počíta sčítaním týchto dvoch hodnôt.

- Ukážky (vysvetlené v sekcii 2.2.7). V nastaveniach herného módu je možné nastaviť počet ukážok medzi žiadnou a šiestimi.
- Hold (vysvetlené v sekcii 2.2.9). V nastaveniach herného módu je možné ju zakázať, povoliť normálne, alebo povoliť nekonečný hold.
- Hard drop (vysvetlené v sekcii 2.2.10).
- Tieňový dielik (vysvetlené v sekcii 2.2.11). V globálnych nastaveniach je možné nastaviť, či tieňový dielik má byť zafarbený ako aktívny dielik.
- Combo (vysvetlené v sekcii 2.2.13),
- Back-to-back (vysvetlené v sekcii 2.2.12). Ako back-to-back vyčistenia sú brané tie vyčistenia, ktoré vyčistia aspoň štyri riadky, a vyčistenia dosiahnuté pomocou spinov (vysvetlené v sekcii 2.2.18).
- Mechaniky pre ukončenie hry (vysvetlené v sekcii 2.2.16). V nastaveniach herného módu je možné si vybrať, aké mechaniky majú byť používané.
- All clear (vysvetlené v sekcii 2.2.19).

3.3.2 Soft drop

Základ tejto mechaniky je spomenutý v sekcii 2.2.1. Štandardne je podporované nastavenie soft drop faktoru na ľubovoľnú hodnotu medzi jedna a pozitívnym nekonečnom.

V tejto bakalárskej práci je soft drop rozšírený o nasledujúce možnosti:

- Oneskorenie soft dropu – hneď po stlačení tlačidla sa aktívny dielik vždy posunie o jeden blok dole, avšak zrýchlenie gravitácie nastane až po uplynutí tohoto intervalu. Toto umožňuje hráčom posúvať aktívny dielik dole blok po bloku aj pri veľmi vysokých hodnotách soft drop faktoru, ako napríklad nekonečno.
- Minimálna gravitácia pri použití soft dropu – pokiaľ je normálna gravitácia hry veľmi nízka a hodnota soft drop faktoru tiež nie je veľmi vysoká, táto hodnota môže pomôcť hráčovi zvýšiť gravitáciu na určitú minimálnu hodnotu. Gravitácia použitá v hre však nikdy nebude nižšia ako normálna gravitácia hry.
- Maximálna gravitácia pri použití soft dropu – pokiaľ je normálna gravitácia hry veľmi vysoká, táto hodnota môže pomôcť hráčovi zvýšiť gravitáciu na určitú maximálnu hodnotu a nie vyššie. Gravitácia použitá v hre však nikdy nebude nižšia ako normálna gravitácia hry. Táto hodnota nemôže byť nižšia ako predchádzajúca hodnota.
- Základ zrýchlenia pri nulovej gravitácii – v niektorých hrách sa môže stať, že normálna gravitácia bude nulová, avšak hráči stále môžu potrebovať použiť soft drop. Toto je hodnota, ktorá sa pri takých okolnostiach použije ako základ násobenia pre soft drop faktor.

Všetky tieto možnosti sú hráčom nastaviteľné v globálnych nastaveniach hry a zároveň je ich možné vynútiť v herných nastaveniach jednotlivých herných módov.

3.3.3 Skórovanie a úrovne hier

Základ týchto mechaník je spomenutý v sekciách 2.2.4 a 2.2.5.

V blockstacker hrách býva väčšinou len jeden systém na skórovanie a určovanie úrovní hry. V rámci tejto bakalárskej práce sú natívne podporované tri rôzne systémy počítania skóre a je možné pomocou užívateľského skriptovania napísať vlastný skórovací systém.

Tento systém sa dá nastaviť v nastaveniach herného módu pod názvom „Game Manager“. Game Managery, ktoré sú súčasťou samotnej hry:

- Classic Game Manager – tento systém sa snaží čo najbližšie napodobniť systém skórovania a úrovní v hre Tetris pre NES. Informácie pre implementovanie tohoto Game Managera boli zobrané z Tetris Wiki, zo stránky Tetris (NES, Nintendo) [1].
- Modern Game Manager without Levelling – tento systém sa snaží čo najbližšie napodobniť systém skórovania v hre Jstris. Keďže Jstris nepodporuje koncept úrovní, tak tento systém tiež úrovne nepodporuje a gravitácia aj činiteľ skórovania zostávajú vždy rovnaké. Informácie pre implementovanie tohoto Game Managera boli získané z vlastných skúseností hraním hry Jstris.
- Modern Game Manager with Levelling – tento systém je podobný ako predchádzajúci, avšak pridáva do hry koncept úrovní. Pri tomto systéme sa každých päť vyčistených riadkov zdvihne úroveň o jednu a toto spôsobí inkrementovanie činiteľa skórovania a takisto 1.5-násobné zvýšenie gravitácie.

  Prvou úrovňou tohoto systému je úroveň 1 s gravitáciou $0.01\overline{6}$ blokov za sekundu.

Okrem vstavaných Game Managerov je možné vybrať v nastaveniach vlastného herného módu možnosť vlastného Game Managera. V tomto prípade si vlastný Game Manager skript dokáže zaregistrovať obslužné funkcie na jednotlivé udalosti, ktoré sa v hre dejú a pridávať skóre, určovať gravitáciu a lock delay (vysvetlené v sekcii 2.2.14).

Presný spôsob, akým je možné písať vlastné Game Manager skripty je uvedený v prílohe, v sekcii A.6.

3.3.4 DAS

Základ tejto mechaniky je spomenutý v sekcii 2.2.6. Štandardne je podporované nastavenie DAS a ARR intervalov.

V tejto bakalárskej práci je DAS rozšírené o nasledujúce možnosti:

- Správanie sa pri stlačení oboch tlačidiel pre pohyb – dá sa nastaviť na pozastavenie oboch smerov DAS, alebo na pozastavenie prvého smeru DAS, alebo ponechanie oboch smerov. Pri nastavení na ponechanie oboch smerov má vždy prednosť smer neskôr stlačeného tlačidla.

  Po pozastavení smeru je pred opätovným spustením DAS potrebné znovu počkať, kým neprejde DAS interval.

  Toto nastavenie pomáha hráčom hýbať dielikmi tak, ako čakajú, že sa budú hýbať. Najintuitívnejšie nastavenie je pre každého hráča rôzne, preto hra poskytuje 3 rôzne možnosti.

- Udalosti, po ktorých je DAS oneskorené – dá sa nastaviť na žiadne, oneskorenie po rotácii, oneskorenie po položení dielika, alebo oboje.

  Toto nastavenie môže pomôcť hráčom, ktorí zvyknú držať tlačidlo pre pohyb príliš dlho po položení, rotácii alebo oboch a nepočítajú s tým, že sa ich nový aktívny dielik začne hýbať hneď po vykonaní týchto akcií.

- Interval DAS oneskorenia – interval medzi akciou, ktorá DAS oneskorí a opätovným spustením DAS.

  Po udalostiach, ktoré podľa predchádzajúceho nastavenia oneskorujú DAS, je automatický pohyb pozastavený až do chvíle, čo tento interval neuplynie. Po tomto pozastavení netreba znovu čakať na DAS interval.

- Možnosť zrušenia DAS oneskorenia stlačením tlačidla pre pohyb – ak je táto možnosť nastavená, DAS oneskorenie je možné zrušiť tým, že hráč opätovne stlačí tlačidlo pre pohyb. Ak nie je nastavená, tak je DAS po danej udalosti kompletne zrušené až kým neuplynie interval oneskorenia.

Všetky tieto možnosti sú hráčom nastaviteľné v globálnych nastaveniach hry a zároveň je ich možné vynútiť v herných nastaveniach jednotlivých herných módov.

Možnosť oneskorenia DAS je inšpirovaná podobnou možnosťou hry TETR.IO.

Možnosť vybrania rôzneho správania pri stlačení oboch tlačidiel pre pohyb je inšpirovaná podobnou možnosťou hry Tetris Effect.

3.3.5 Náhodné generátory

Základ tejto mechaniky je spomenutý v sekcii 2.2.8.

V rámci tejto bakalárskej práce je súčasťou hry niekoľko štandardných náhodných generátorov. Pomocou užívateľského skriptovania je taktiež možné napísať vlastné náhodné generátory.

Generátory, ktoré sú súčasťou hry:

- Generátor s jednoduchým obmedzením opakovania.
- Generátor s pokročilým obmedzením opakovania.
- Generátor 7-bag.
- Generátor 14-bag. Toto je obdoba predchádzajúceho generátora, kde sa do „vrecka“ na začiatku pridajú dva kusy každého typu dieliku, nie len jeden.
- Generátor párov – tento generátor funguje úplne náhodne, ale miesto generovania po jednom kúsku mení typ vygenerovaného kúsku vždy po dvoch.
- Úplne náhodný generátor.

Okrem vstavaných náhodných generátorov je v nastaveniach herného módu aj možnosť pridania vlastného generátora. V tomto prípade vlastný generátor potrebuje implementovať dve funkcie – funkciu na generovanie náhodných kúskov a funkciu na reštart svojho stavu.

Presný spôsob vytvárania a používania vlastných náhodných generátorov je popísaný v prílohe, v sekcii A.8.

3.3.6 Uzamykanie dielikov

Základ tejto mechaniky je spomenutý v sekciách 2.2.14 a 2.2.15.

Pokladanie alebo uzamykanie dielikov v tejto bakalárskej práci môže nastať niekoľkými rôznymi spôsobmi.

1. Uzamknutie dielika po vypršaní intervalu lock delay – tento interval je nastaviteľný v nastaveniach vlastného herného módu. Takisto sa dá nastaviť, od akej udalosti sa začína počítať. Buď je možné tento interval počítať od momentu, čo sa kúsok dotkne zeme, alebo od momentu, čo sa kúsok kvôli gravitácii pokúsi pohnúť dole a nebude to možné.

   V hrách bez mechaniky hard drop (vysvetlené v sekcii 2.2.10) je toto hlavný spôsob uzamknutia dielika.

2. Uzamknutie dielika po použití mechaniky hard drop – ak je v nastaveniach herného módu povolená mechanika hard drop, tak sa stlačením hard drop tlačidla aktívny dielik automaticky posunie na najnižšie možné miesto a okamžite uzamkne.

3. Uzamknutie dielika po aktivácii mechaniky hard lock – u mechaniky hard lock sú dostupné nastavenia typu a počtu. Typ hard locku určuje, ktoré udalosti budú inkrementovať počet.

   V nastaveniach herného módu je možné vybrať typ hard locku z limitovaného času, limitovaných pohybov či rotácií, limitovaného počtu stlačení pohybových a rotačných tlačidiel, alebo na nekonečný pohyb. V prípade nekonečného pohybu sa mechanika hard lock nikdy neaktivuje.

   Aktívny dielik sa uzamkne, ak počet ubehnutých udalostí od dotknutia zeme dosiahne hodnotu špecifikovanú v nastaveniach herného módu. V prípade limitovaného času sa používa počet ubehnutých sekúnd.

3.3.7 Rotačné systémy

Základ tejto mechaniky je spomenutý v sekcii 2.2.17.

Táto bakalárska práca podporuje dva vstavané rotačné systémy a zároveň umožňuje hráčom napísať vlastný rotačný systém.

Vlastný rotačný systém je možné vybrať v nastaveniach herného módu. Vo vlastnom rotačnom systéme je možné popísať všetky použité kicky pre prechod z akéhokoľvek rotačného stavu do akéhokoľvek iného rotačného stavu. Takisto je možné definovať, aké kicky sa budú brať ako výnimky pre detekovanie spinov, a rotačný stav, v ktorom sa dielik objaví.

Presný spôsob, ako vytvoriť a používať vlastné rotačné systémy je uvedený v prílohe, v sekcii A.9.

3.3.8 Spiny

Základ tejto mechaniky je spomenutý v sekcii 2.2.18.

Vo väčšine blockstacker hier sú podporované jedine T-Spiny podľa pravidiel vysvetlených vyššie. V rámci tejto bakalárskej práce je však možné v nastaveniach herného módu vybrať, či sa majú detekovať T-Spiny, spiny všetkých dielikov, „hlúpe spiny“, alebo žiadne spiny.

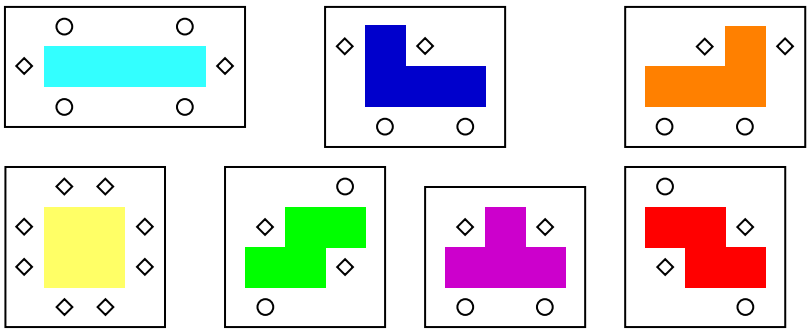
V prípade detekcie T-Spinov pravidlá zostávajú rovnaké ako vysvetlené vyššie.

Pre detekciu spinov všetkých dielikov boli pravidlá T-Spinov zovšeobecnené na použitie pre akýkoľvek dielik. Takto vznikol koncept detektorov spinov, ktoré sú špecifikované pre každý typ dielika. Tieto detektory sú miesta relatívne ku dieliku, na ktorých sa zisťuje, či je dané miesto zabrané alebo nie. Ak je miesto detektoru zabrané, tak sa daný detektor berie ako „vyplnený“.

Pre každý typ dielika je zároveň definovaný minimálny počet detektorov, ktoré musia byť vyplnené pre detekciu spinu. U T dielikov sú toto tri zo štyroch detektorov.

Niekoľko z detektorov je zároveň označených ako „detektory plných spinov“. Tieto detektory musia byť všetky vyplnené, aby sa spin počítal ako plný. Ak je vyplnené dostatočné množstvo detektorov na detekciu spinu, ale nie sú vyplnené všetky detektory plných spinov, tak sa daný spin berie ako spin mini.

Polohy detektorov spinov, ako sú implementované v hre, sú zobrazené na obrázku 3.3.



Obr. 3.3: Obrázok znázorňuje rozloženie „detektorov spinov“ pre každý zo štandardných 7 dielikov. Obyčajné detektory sú označené kruhmi a detektory plných spinov sú označené štvorcami. Každý dielik je navyše obtiahnutý čiarou, čo určuje, ktoré detektory danému dieliku patria.

V prípade detekcie „hlúpych spinov“ sa za spin považuje akékoľvek položenie dielika, pred ktorým bola ako posledná akcia vykonaná rotácia.

3.3.9 Generovanie odpadových riadkov

Základ tejto mechaniky je spomenutý v sekcii 2.2.21.

V rámci tejto bakalárskej práce je možné v nastaveniach vlastného herného módu definovať, akým spôsobom sa majú generovať odpadové riadky. Pokiaľ je toto nastavenie nedostatočné, tak je ďalej možné pomocou užívateľského skriptovania napísať vlastný generátor odpadových riadkov.

Vstavané možnosti pre generovanie odpadových riadkov sú preferovaná výška odpadu v hre a typ generovania. Existuje päť rôznych vstavaných typov generovania:

- Generovanie dier vysokých jeden blok,
- generovanie dier vysokých jeden alebo dva bloky,
- generovanie dier vysokých jeden, dva alebo tri bloky,
- generovanie dier vysokých jeden, dva, tri alebo štyri bloky.

V prípade použitia vlastného generátoru odpadových riadkov skript pre generovanie potrebuje podporovať dve funkcie – funkciu pre generovanie odpadových riadkov a funkciu pre resetovanie stavu generátoru.

Presný spôsob pre vytváranie a používanie vlastných generátorov odpadových riadkov je popísaný v prílohe, v sekcii A.7.

3.3.10 Prevencia neúmyselných hard dropov

Táto mechanika bola inšpirovaná podobnou mechanikou v hre TETR.IO.

V nastaveniach hry TETR.IO sa toto dá nastaviť na zapnuté alebo vypnuté. Mechanika funguje tak, že po uzamknutí dielika je dočasne zneprístupnené tlačidlo pre hard drop.

Toto však funguje len v prípade, že kúsok bol uzamknutý kvôli uplynutiu intervalu lock delay alebo kvôli použití mechaniky hard lock, ktoré sú dostupné aj v hre TETR.IO. Ak hráč jednoducho veľmi rýchlo za sebou stlačí tlačidlo pre hard drop, tak táto mechanika druhému hard dropu nezabráni.

Okrem toho nie je v hre vysvetlené, aký dlhý interval neprístupnosti tlačidla reálne je. Je definovaný len ako „niekoľko snímkov“.

Mechanika prevencie neúmyselných hard dropov v tejto bakalárskej práci funguje na podobnom princípe. V nastaveniach ovládania hry si hráč môže definovať, aký dlhý má byť interval neprístupnosti tlačidla pre hard drop po položení dielika. Ak je tento interval nenulový, tak bude po dobu od položenia dielika až do uplynutia intervalu hard drop tlačidlo neprístupné, nezáležiac na tom, ako bol dielik položený.

3.3.11 Diagonálne uzamknutie

Táto mechanika bola prebraná z hry Tetris Effect a mierne rozšírená.

V nastaveniach hry Tetris Effect sa toto dá nastaviť na zapnuté alebo vypnuté. Mechanika funguje tak, že ak hráč momentálne drží tlačidlo pre pohyb doprava alebo doľava, tlačidlá pre hard drop a soft drop sú neprístupné.

V tejto bakalárskej práci je možné si vybrať, či má táto mechanika preferovať horizontálny alebo vertikálny pohyb, alebo byť vypnutá.

Ak je vybrané preferovanie horizontálneho pohybu, tak funguje rovnako ako v hre Tetris Effect.

Ak je vybrané preferovanie vertikálneho pohybu, tak pri držaní tlačidla pre soft drop sú tlačidlá pre pohyb doprava a doľava neprístupné.

3.3.12 Initial Actions

Táto mechanika bola prebraná z mechaník hry Tetris Effect pod názvami Initial Rotation a Initial Hold.

Tieto mechaniky umožňujú použiť mechaniku hold alebo mechaniky rotácií ešte predtým, ako sa dielik objaví na hernej ploche.

V nastaveniach hry Tetris Effect sa tieto mechaniky dajú osobitne nastaviť na zapnuté alebo vypnuté. Ďalej je k nim prístupné nastavenie typu funkcionality.

Pri type A sa mechaniky aktivujú, ak sú tlačidlá pre rotáciu alebo hold práve držané, keď sa objaví nový aktívny dielik.

Pri type B sa mechaniky aktivujú iba ak boli príslušné tlačidlá stlačené počas intervalu piece spawn delay (vysvetlené v sekcii 2.2.2).

V tejto bakalárskej práci sú k dispozícii rovnaké nastavenia tejto mechaniky ako v hre Tetris Effect.

Mechanika Initial Rotation vždy vyskúša len prvú možnú polohu dielika po otočení. Toto umožňuje kúsok otáčať pomocou tejto mechaniky inak v rôznych rotačných systémoch, ale neposkytuje to veľké zjednodušenie prežitia.

Túto mechaniku je zároveň v nastaveniach vlastného herného módu možné vypnúť.

3.3.13 Automatické použitie Initial Rotation

Táto mechanika bola navrhnutá ako alternatíva mechaniky Initial Rotation z predchádzajúcej sekcie a podľa znalostí autora tejto práce je v dobe písania práce unikátna.

Mechanika spočíva v tom, že ak nie je možné nový dielik pri objavení umiestniť na hernú plochu, tak sa dielik pred objavením automaticky pokúsi otáčať. Bude sa otáčať dovtedy, kým nenájde polohu, v ktorej sa na hernú plochu bude môcť umiestniť, alebo kým nezistí, že taká poloha neexistuje.

Pri každom otočení sa použije vždy len prvý kick z aktívneho rotačného systému (vysvetlené v sekcii 2.2.17), podobne ako u Initial Actions. V nastaveniach ovládania hry sa táto mechanika dá nastaviť na jednu z troch hodnôt:

- Vypnutá.
- Rotovanie po smere hodinových ručičiek – pri tejto hodnote sa aktívny dielik pokúsi najprv otočiť po smere hodinových ručičiek, potom o 180 stupňov a potom proti smeru hodinových ručičiek.
- Rotovanie proti smeru hodinových ručičiek – pri tejto hodnote sa aktívny dielik pokúsi najprv otočiť proti smeru hodinových ručičiek, potom o 180 stupňov a potom po smere hodinových ručičiek.

Pokus o rotáciu o 180 stupňov bude vykonávaný len v prípade, že nastavenia herného módu túto rotáciu povoľujú.

V prípade, že hráč použil Initial Rotation na rotáciu dielika pred objavením, ale dielik stále nemá miesto pre objavenie sa, je stále možné prežiť, ak automatická rotácia nájde polohu, v ktorej by sa dielik objaviť mohol.

Túto mechaniku je zároveň v nastaveniach vlastného herného módu možné vypnúť.

3.3.14 Posun aktívneho dielika ku stene

Táto mechanika bola navrhnutá ako alternatíva používania DAS. Podľa znalostí autora tejto práce je v dobe písania práce unikátna.

Mechanika spočíva v tom, že stlačením tlačidla pre posun aktívneho dielika k stene sa dielik okamžite pohne ku stene bez toho, aby hráč musel čakať na uplynutie DAS intervalu a nasledovných ARR intervalov.

Mechanika má nevýhodu v tom, že pri jej používaní hráč potrebuje používať viac tlačidiel a tým značne komplikuje ovládanie hry, ale zároveň ovládanie hry potenciálne zrýchľuje pre hráčov, ktorí danú mechaniku dokážu efektívne používať.

Túto mechaniku je v nastaveniach vlastného herného módu možné vypnúť.

3.3.15 Odrezávanie dielikov nad určitou výškou

Táto mechanika bola prebraná z hry Jstris a mierne rozšírená.

V hre Jstris sú všetky bloky nad výškou 21 automaticky vymazané. Herné pole v hre Jstris je vždy vysoké presne 20 blokov a toto sa nedá zmeniť, preto dáva zmysel mať túto hodnotu nastavenú na pevno.

V tejto bakalárskej práci je však možné vo vlastných herných nastaveniach zvoliť výšku herného poľa na ľubovoľnú hodnotu od 2 do 400 blokov, preto je možné takisto upraviť výšku „odrezávania“ dielikov na ľubovoľnú hodnotu od 10 do 420 blokov.

3.4 Prispôsobenie výzoru, zvuku a počítania štatistík

Prispôsobenie výzoru a zvuku je v tejto bakalárskej práci rozdelené do prispôsobenia herných pozadí, prispôsobenia herného zvuku a prispôsobenia výzoru blokov jednotlivých dielikov. Výzor užívateľského rozhrania nie je prispôsobiteľný. Okrem toho je takisto možné upravovať živé počítadlá štatistík, ktoré hráčovi zlepšujú prehľad o hre počas hrania.

V tejto sekcii budú postupne popísané jednotlivé systémy prispôsobenia hry a ich funkcionalita.

3.4.1 Počítanie štatistík

V tejto bakalárskej práci je pre hry vstavané počítanie východzích štatistík, ktoré sa zároveň ukladajú do súborov prehrávok. Počas hry sú takisto zobrazované živé počítadlá, ktoré zobrazujú priebeh štatistík v reálnom čase.

Každý herný mód má vstavaný zoznam živých počítadiel, ktoré sa v ňom zobrazujú ako východzie. Živé počítadlá štatistík je však možné pomocou užívateľského skriptovania naprogramovať na počítanie akýchkoľvek štatistík.

Takto naprogramované počítadlá štatistík je možné v nastaveniach hry zoskupiť do skupín a v herných menu jednotlivých herných módov je možné vybrať preferovanú skupinu.

Do skupín živých počítadiel je okrem vlastných počítadiel možné pridať aj vstavané živé počítadlá. Tieto vstavané počítadlá sú:

- Stopwatch – počíta ubehnutý čas.
- Lines Cleared – počíta vyčistené riadky.
- Pieces – počíta položené dieliky a rýchlosť, akým sa dieliky pokladajú.
- Score – počíta skóre a priemerné skóre na počet položených dielikov.
- Inputs – počíta stlačené tlačidlá.
- Level – ukazuje úroveň a priebeh podmienky pre zmenu úrovne.
- all clears – počíta vykonané all cleary.
- Holds – počíta vykonané holdy.
- Remaining End Condition – počíta podmienku pre ukončenie hry.

Okrem týchto počítadiel je rovnakým spôsobom vytvorený aj takzvaný „action text“. Tento action text slúži na zobrazovanie vykonaných akcií v čase, kedy nastanú. Pre príklad,

ak užívateľ vykoná T-Spin (vysvetlené v sekcii 2.2.18), tak action text ohlási, že nastal T-Spin. Toto môže hráčom pomôcť identifikovať ako funguje detekcia T-Spinov bez nutnosti čítania dokumentácie.

Action text je rozdelený do štyroch rôznych počítadiel:

- Clear Type Action Text – zobrazuje sa, keď nastane akékoľvek vyčistenie.
- All clears Action Text – zobrazuje sa, keď nastane all clear.
- Back-to-back Action Text – zobrazuje sa, keď nastane back-to-back vyčistenie.
- Combo Action Text – zobrazuje momentálny stav comba.

Vlastné živé počítadlá štatistík alebo časti action textu fungujú na podobnom princípe ako vlastné Game Managery (vysvetlené v sekcii 3.3.3). Každé vlastné počítadlo si dokáže zaregistrovať obslužné funkcie na jednotlivé udalosti v hre a následne v rámci obsluhy udalostí nastavovať svoj zobrazený text, jeho farbu a viditeľnosť. Skripty počítadiel majú takisto prístup k niekoľkým funkcionalitám pre zjednodušenie ich programovania:

- Hodnoty vstavaných štatistík, ktoré hra počíta automaticky,
- momentálny čas hry,
- momentálny stav hernej plochy,
- úžitkové funkcie, ktoré slúžia k zjednodušeniu formátovania čísiel alebo času do textových reťazcov.

Počítadlá okrem obslužných funkcií na udalosti v hre takisto dokážu špecifikovať funkciu, ktorá sa bude volať periodicky každý časový interval. Dĺžka tohoto intervalu sa dá nastaviť v nastaveniach počítadla, zároveň s veľkosťou a polohou daného počítadla.

Presný spôsob, akým je možné vytvárať a používať vlastné počítadlá štatistík je vysvetlený v prílohe, v sekcii A.10.

3.4.2 Prispôsobenie herných pozadí

Systém prispôsobenia herných pozadí je zo všetkých systémov prispôsobenia najjednoduchší. V herných nastaveniach je možné vybrať si kolekciu pozadí, podľa kolekcií, ktoré má užívateľ prístupné v perzistentnom úložisku.. Po výbere kolekcie pozadí sa automaticky pozadia načítajú a aplikujú.

Každé pozadie môže byť buď jeden obrázok vo formáte `jpg` alebo `png`, alebo skupina takýchto obrázkov. Ak sa ako pozadie použije skupina obrázkov, tak sa pri načítaní pozadia náhodne vyberie jeden zo špecifikovaných obrázkov.

Presný spôsob, akým je možné používať vlastné herné pozadia je vysvetlený v prílohe, v sekcii A.3.

3.4.3 Prispôsobenie výzoru blokov

Systém prispôsobenia výzoru blokov je komplexný systém na nastavenie výzoru jednotlivých blokov herných kúskov a hernej mriežky. V herných nastaveniach je možné vybrať si použitú konfiguráciu výzoru blokov.

Konfigurácia výzoru obsahuje zoznam nastavení, ktoré umožňujú špecifikovať ako presne majú všetky bloky pri použití daného výzoru vyzerať. Každý typ bloku môže mať priradených niekoľko záznamov o častiach výzoru. Záznamy poskytujú nasledujúce možnosti:

- Vybranie typu dieliku – môže byť buď typ daného dielika podľa jeho názvu, alebo jeden z ďalších špecifických typov blokov, ktoré sa v hre vyskytujú. Medzi tieto typy patria dielik odložený hold mechanikou, tieňový dielik, varovný dielik, blok odpadových riadkov alebo herná mriežka.
- Vybranie blokov dielika, na ktoré sa má výzor použiť – toto je možné vybrať len pre obyčajné dieliky.
- Vybranie vrstvy výzoru – v prípade, že daný blok má priradených viac záznamov, bude číslo vrstvy použité na rozhodovanie poradia vykresľovania.
- Možnosť zobrazovať skin ako prepojený – ak je daný výzor označený ako prepojený, tak bude hra zisťovať na ktorých stranách daného bloku v hre sú prítomné iné bloky a podľa toho vyberať konfiguráciu výzoru. Ak nie, tak bude pevne vybraná jedna sekvencia obrázkov.
- Možnosť otáčania výzoru bloku s dielikom – toto je možné vybrať len pre obyčajné dieliky.
- Možnosť špecifikovania rýchlosti animácie výzoru, ak má byť výzor animovaný.
- Možnosť vybrania presných častí obrázku – výzor bloku môže byť buď jeden, alebo sekvencia obrázkov. V obidvoch prípadoch je možné špecifikovať, aký súbor sa používa ako zdroj obrázku a odkiaľ presne v súbore sa má daný obrázok vyrezať.

Prepojené výzory používajú konfigurácie, ktoré fungujú na základe hrán a rohov. Každá konfigurácia má zoznam hrán a rohov, ktoré sú na jej obrázkoch alebo obrázku. Podľa tohto zoznamu hra následne vyberá, aká konfigurácia sa má použiť.

Výhodou tohoto systému je veľká škála možných výzorov a voľnosť v organizovaní súborov s obrázkami, ale nevýhodou je, že vytvorenie konfigurácie je pre komplikovanejšie výzory extrémne zdĺhavé a nepraktické, pokiaľ tento proces tvorca výzoru nie je schopný automatizovať.

Presný spôsob, akým je možné vytvárať a používať vlastné výzory blokov, je vysvetlený v prílohe, v sekcii A.4.

3.4.4 Prispôsobenie zvuku

Systém prispôsobenia zvuku obsahuje možnosti prispôsobenia hernej hudby v rôznych častiach hry a takisto možnosť prispôsobenia zvukových efektov. Pomocou užívateľského skriptovania je takisto možné naprogramovať, kedy presne sa ktoré zvuky majú prehrávať. V herných nastaveniach je možné si vybrať balíček hudby a zvukových efektov. Podľa nastavení v tomto balíčku sa následne prispôsobí herná hudba a zvukové efekty.

Hudba je rozdelená na hudbu pre menu, hudbu pre herné nastavenia a hudbu pre hru. Herná hudba môže byť ďalej rozdelená do skupín, čo umožňuje hráčom vybrať si pri rôznych herných módoch rôzne skupiny hudby podľa toho, akú majú zrovna náladu. V herných nastaveniach je možné vybrať buď jeden konkrétny hudobný klip, alebo skupinu klipov definovanú v hudobnej konfigurácii, alebo nechať hru vybrať úplne náhodný hudobný klip z dostupnej hudby.

Vlastné zvukové efekty je možné buď jednoducho použiť na nahradenie východzích zvukových efektov v hre, alebo je možné pre úplnú kontrolu napísať skript, ktorý bude dostávať upozornenia na udalosti v hre a podľa nich prehrávať rôzne zvukové efekty.

Presný spôsob, akým je možné prispôsobovať hernú hudbu a zvukové efekty je uvedený v prílohe, v sekcii A.5.

Kapitola 4

Použité technológie

Táto kapitola rozoberá herné engine ako také a následne herný engine Unity, ktorý bol použitý pre vytvorenie tejto bakalárskej práce. Taktiež je tu spomenutý jazyk Lua a dôvody, prečo bol vybraný ako skriptovací jazyk pre tento projekt.

4.1 Herné engine

Herné engine je typ software, ktorý slúži k vývoji hier. Aplikácie hier majú mnohé rôzne požiadavky, ktoré nie sú potrebné pre iné druhy aplikácií, ako napríklad vykresľovanie 3D prostredia, správa rôznych typov zdrojov, simulácia fyziky v reálnom čase, animácia, presné spracovanie užívateľského vstupu a veľká kontrola prehrávaného audia.

Herný engine je prostredie, ktoré splňuje všetky tieto požiadavky a poskytuje rozhranie pre organizovanie jednotlivých častí hry. Takisto poskytuje užívateľom možnosť napísať vlastné skripty pre pravidlá hry, ktoré budú ovládať detaily samotnej hernej funkcionality.

Dnes existuje niekoľko rôznych herných engine, ktoré sú cielené na rôzne skupiny ľudí podľa toho, na aký typ hier sú špecializované. Mnohé herné engine sú však napísané tak, aby ich bolo možné použiť na čo najväčšiu škálu rôznych hier. Medzi dnes populárne herné engine patria:

- Unity – herný engine použitý na túto bakalársku prácu. Podporuje vytváranie 2D aj 3D hier pre viac než 20 rôznych platforiem a je známy ako veľmi všeobecný a rozšírený herný engine. Okrem hier je možné v ňom vytvárať aj filmy a animácie alebo aplikácie, ktoré potrebujú využívať 3D grafiku. Vďaka svojej jednoduchosti a prívetivej monetizácii je dnes najpoužívanejším herným engine.

  Ďalej je rozobraný v sekcii 4.1.1.

- Unreal Engine – tento engine je známy hlavne kvôli svojím vynikajúcim grafickým schopnostiam, vďaka ktorým je možné v ňom vytvárať nádherné prostredia v realistickom štýle s minimálnou námahou. Je cielený hlavne na veľké štúdiá, ktoré sú schopné využiť tieto grafické schopnosti. Okrem hier je často používaný aj pre vizuálne efekty v živých filmoch.

- Godot – extrémne malý a ľahký game engine, ktorý je možné spustiť aj na mobiloch či dokonca v prehliadači. Godot Game Engine je takisto plne open-source a jeho zdrojový kód je dostupný pod licenciou MIT, čo dovoľuje vývojárom ho akokoľvek upravovať a prípadne aj upravené verzie predávať.

Godot cieli hlavne na malých vývojárov, ktorí nepotrebujú veľké a ťažkopádne engine ako Unreal, ale stačí im jednoduché a rýchle prostredie. Takisto je jeho veľkou výhodou, že akékoľvek v ňom vytvorené hry sú pre vývojárov úplne zadarmo.

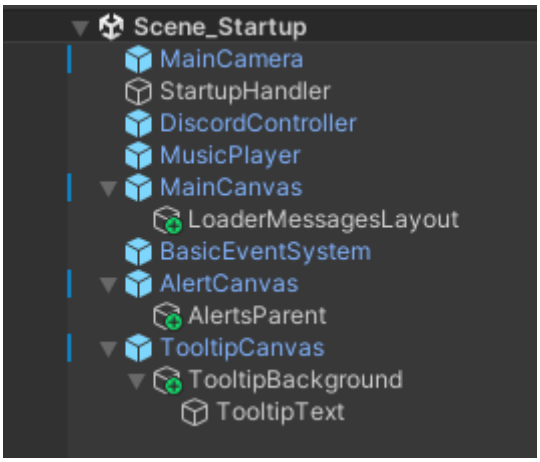
4.1.1 Herný engine Unity

Herný engine Unity je momentálne najpopulárnejším herným engine pre vývoj 2D a 3D hier. Medzi známe hry vytvorené v Unity patria napríklad Genshin Impact[1], Fall Guys[2], Cuphead[3] a Hollow Knight[4].

Informácie v tejto sekcii boli prebrané a voľne preložené z Unity manuálu [3].

Základné stavebné bloky hier v Unity sú scény. Každá scéna reprezentuje nejakú časť hry. Dané scény ďalej obsahujú takzvané „Game Objecty" v hierarchickej štruktúre. Každý objekt, ktorý sa v Unity hre nachádza, je Game Objectom. Tieto Game Objecty môžu obsahovať kamery, obrázky, 3D objekty, zdroje svetla, alebo inú funkcionalitu. Game Objecty môžu byť umiestnené priamo v scéne, alebo môžu byť priradené pod ďalšie Game Objecty. Každá scéna je teda stromovým grafom, kde koreňom stromu je samotná scéna a ďalšie uzly sú Game Objecty.

Príklad stromovej štruktúry scény je zobrazený na obrázku 4.1.



Obr. 4.1: Obrázok znázorňuje hierarchické usporiadanie scény v hernom engine Unity. Scene_Startup – scéna – je koreňom hierarchie a obsahuje niekoľko Game Objectov. Niektoré z Game Objectov znova obsahujú ďalšie detské Game Objecty.

Samotný Game Object však neposkytuje žiadnu funkcionalitu, ale slúži pre obsiahnutie komponentov. Komponenty sú časti Game Objectu a tvoria funkcionalitu hry. Užívateľ herného engine Unity dokáže Game Objectom priraďovať vstavané komponenty, ktoré sú súčasťou samotného engine, alebo implementovať vlastné komponenty v skriptovacom jazyku Unity – C#. Komponenty je takisto možné konfigurovať špecificky pre každý Game Object pomocou verejných polí, ktoré sú exportované do Unity editora. Každý Game Object navyše vždy obsahuje komponent typu Transform, ktorý sa nedá odobrať a reprezentuje polohu, rotáciu a mierku daného Game Objectu.

[1] https://genshin.hoyoverse.com/en/
[2] https://www.fallguys.com/en-US
[3] https://www.cupheadgame.com/
[4] https://www.hollowknight.com/

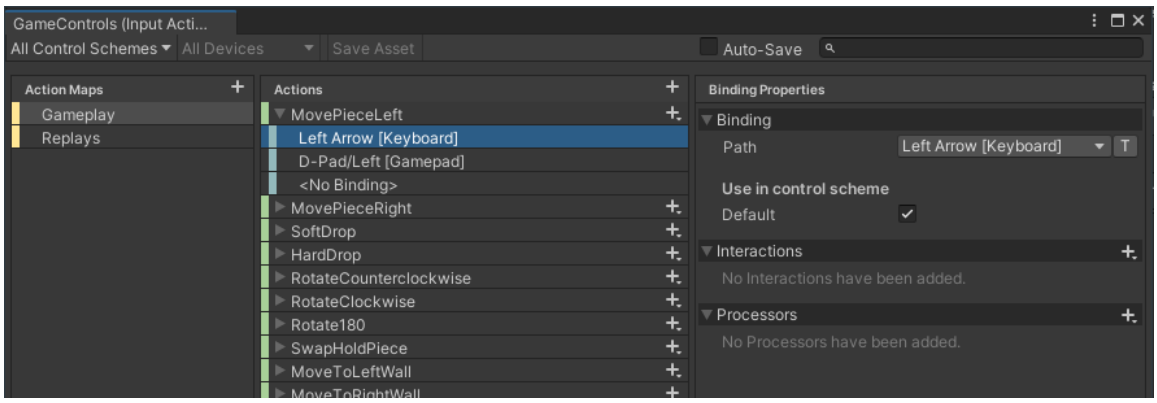
Určitú konfiguráciu Game Objectov a ich komponentov je možné uložiť do súboru a tento súbor následne používať ako základ na vytváranie inštancií. Takéto súbory sa nazývajú Prefaby a je možné vďaka nim jednoducho vytvárať vzory Game Objectov alebo meniť konfiguráciu všetkých inštancií naraz.

Okrem skriptov, ktoré slúžia ako komponenty pre Game Objecty, je možné vytvárať aj skripty pre takzvané „Scriptable Objecty“ – tieto slúžia na ukladanie nastavení a iných dát, ktoré sa zvyčajne počas behu aplikácie nezvyknú meniť.

Ďalšie dôležité časti herného engine Unity sú systém pre správu užívateľského vstupu – Input System, a systém pre vytváranie užívateľských rozhraní – Unity UI.

Input System poskytuje užívateľom Unity možnosť definovať rôzne užívateľské akcie a priradiť k nim tlačidlá, ktoré ich budú ovládať. Takisto podporuje jednoduché menenie priradených tlačidiel počas behu aplikácie a ukladanie zmenených priradení do textového formátu. Na všetky užívateľské akcie je možné následne v kóde reagovať. Veľká výhoda Input Systemu je schopnosť zaznamenať moment, kedy nastala samotná akcia. Vďaka tomu je možné reagovať na užívateľský vstup s extrémnou presnosťou aj pri nízkych obnovovacích rýchlostiach aplikácie.

Užívateľské rozhranie pre prácu s mapami užívateľských akcií je zobrazené na obrázku 4.2.



Obr. 4.2: Obrázok znázorňuje užívateľské rozhranie pre vytváranie máp užívateľských akcií. Na ľavej časti obrázku sú zobrazené jednotlivé mapy, ktoré sú použité v rôznych častiach hry. V strednej časti sú zobrazené jednotlivé akcie a u prvej akcie sú rozbalené tlačidlá, ktoré sú k nej priradené. Nakoniec na pravej časti je zobrazená možnosť vybrať špecifické tlačidlo a prípadne špecifikovať interakcie alebo procesory pre dané priradenie.

Unity UI je štandardný spôsob vytvárania užívateľských rozhraní pre Unity. Je pevne previazaný so systémom Game Objectov. Pri používaní Unity UI je treba pre každú časť užívateľského rozhrania vytvoriť špeciálny Game Object typu Canvas, ktorý musí byť rodičovským Game Objectom všetkých prvkov užívateľského rozhrania. Objekty v tomto Game Objecte majú miesto obyčajného východzieho komponentu Transform komponent RectTransform, ktorý nereprezentuje Game Object ako bod, ale ako obdĺžnik.

4.2 Skriptovací jazyk Lua

Skriptovanie je možnosť písania kódu, ktorý bude v aplikácii vykonávať nejakú funkcionalitu. Pre užívateľské skriptovanie pre túto bakalársku prácu bol vybraný jazyk Lua. V tejto

sekcii budú rozobrané dôvody, prečo bol vybraný práve tento jazyk a jeho výhody. Všetky informácie v tejto sekcii boli prebrané a voľne preložené zo stránky jazyku Lua [2].

Lua je jednoduchý dynamicky typovaný programovací jazyk. Beží pomocou interpretovania kódu na virtuálnom stroji založenom na registroch. Podporuje automatickú správu pamäti pomocou inkrementálnej „garbage collection“.

Medzi výhody jazyka Lua patria:

- Veľmi malá dátová stopa – Lua interpret má približne len 300 KB.
- Veľká rýchlosť – jazyk Lua je všeobecne známy ako jeden z najrýchlejších skriptovacích jazykov na svete.
- Známosť – Lua je použitý v mnohých veľkých aplikáciách a predovšetkým je veľmi často používaný ako skriptovací jazyk pre hry.
- Možnosť používať tento jazyk zadarmo – je dostupný pod licenciou MIT.

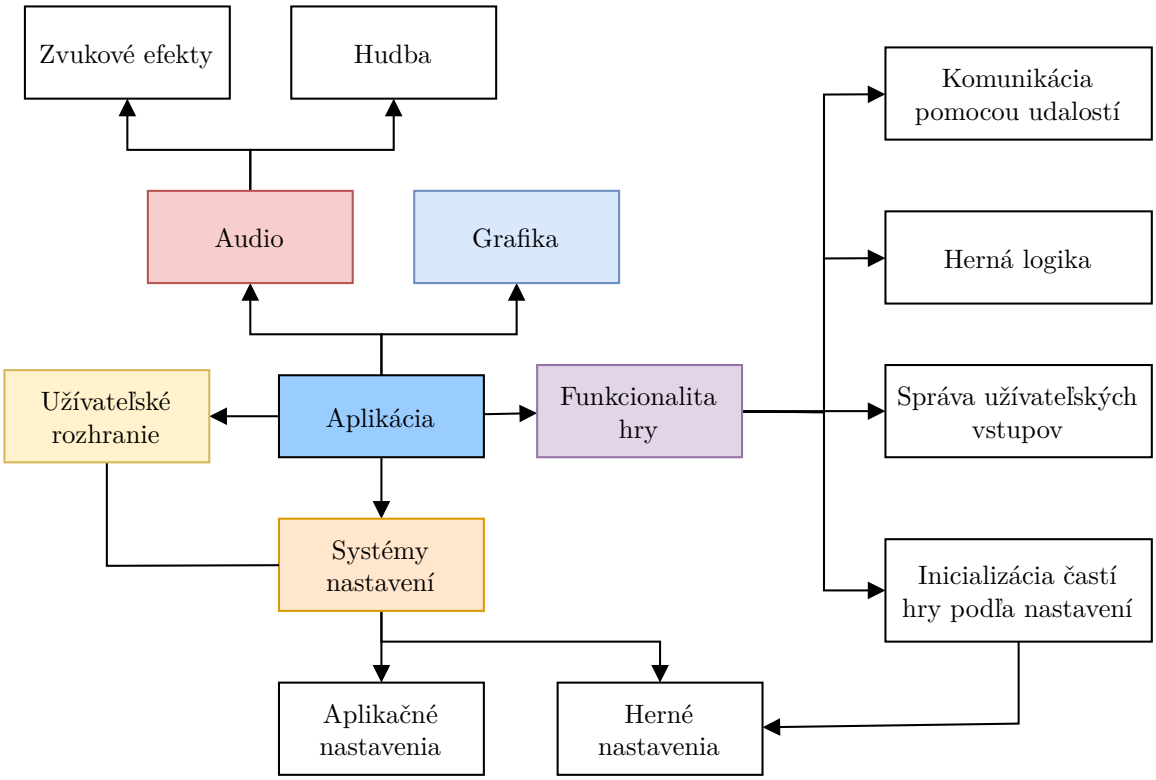
Vďaka týmto výhodám, jeho jednoduchosti a existencii výbornej knižnice NLua[5] pre platformu .NET a jazyk C# bol tento jazyk vybraný ako skriptovací jazyk pre túto bakalársku prácu.

[5] http://nlua.org/

Kapitola 5

Implementácia

V tejto kapitole bude rozobraná implementácia blockstacker hry a jej jednotlivých častí. Tieto časti a ich vzájomné súvislosti sú zobrazené na obrázku 5.1.



Obr. 5.1: Obrázok znázorňuje rozdelenie aplikácie do hlavných častí: audio, obrázky, funkcionalita, systémy nastavení, a užívateľské rozhranie. Audio je ďalej rozdelené na všadeprítomnú hudbu a zvukové efekty v hre. Funkcionalita hry zahŕňa hernú logiku, ktorá určuje chovanie hry, komunikáciu medzi časťami hry, správu užívateľských vstupov a ešte pred začiatkom samotnej hry jej inicializáciu. Nastavenia sú rozdelené na aplikačné nastavenia a herné nastavenia jednotlivých módov, ktoré vplývajú na inicializáciu hry.

- V sekcii 5.1 bude popísaná celková štruktúra projektu na škále scén v hernom engine Unity.

- V sekcii 5.2 budú popísané zdroje pre audio a grafické časti aplikácie a možnosti, ako ich upravovať.
- Sekcia 5.3 rozoberá systémy nastavení a ich prepojenie so zvyškom hry.
- V sekcii 5.4 je popísané užívateľské rozhranie a užívateľský zážitok v aplikácii.
- Nakoniec sekcia 5.5 rozoberá funkcionalitu samotnej hry a jej častí.

5.1 Rozvrhnutie v Unity

V hernom engine Unity je aplikácia rozdelená na 5 rôznych scén:

1. `Scene_Startup` – táto scéna slúži na načítanie herných nastavení a akýchkoľvek prípadných vlastných užívateľských úprav, ktoré sú v nastaveniach špecifikované. Takisto sú v nej umiestnené singleton objekty, ktoré sú aktívne počas celého životného cyklu hry.
2. `Scene_Menu_Main` – táto scéna slúži ako hlavné miesto pre navigáciu v hre. Je tu možné ukončiť aplikáciu, vybrať si preferovaný herný mód, vybrať súbor prehrávky na zobrazenie alebo upraviť akékoľvek globálne nastavenia.
3. `Scene_Menu_GameSettings_Singleplayer` – táto scéna poskytuje možnosť nastaviť jednotlivé časti herného módu, ktoré nemajú na pravidlá hry vplyv, ako odpočet, počiatočný level, hudba a počítadlá štatistík.

   Niektoré nastavenia sú prístupné iba pre určité typy hier, podľa toho, či sú pre ne relevantné.
4. `Scene_Menu_GameSettings_Custom` – táto scéna poskytuje možnosť nastaviť vlastný herný mód a obsahuje všetky dostupné herné nastavenia.
5. `Scene_Game_Singleplayer` – scéna hry. Je spoločná pre všetky herné módy a obsahuje všetko potrebné pre inicializáciu akéhokoľvek z nich.

Singleton Game Objecty, ktoré sú načítané v počiatočnej scéne:

- `DiscordController` – tento objekt sa stará o posielanie upozornení Discordu, aby dokázal ukazovať v akej časti hry sa hráč práve nachádza.
- `MusicPlayer` – objekt slúži pre prehrávanie hudby. Zostáva aktívny počas celého cyklu hry, aby hudba nebola odseknutá pri zmene scén.
- `AlertDisplayer` – objekt slúži pre zobrazovanie upozornení.
- `TooltipCanvas` – tento objekt slúži ako „tooltip“ – ak hráč podrží kurzor myši nad nejakým objektom, ktorý má nastavený pomocný popis, pomocou tohoto objektu sa nad kurzorom myši tento popis zobrazí.

5.2 Audio a grafika

V tejto sekcii budú rozobrané systémy pre audio a grafika hry a uvedené zdroje, odkiaľ boli získané audio klipy a obrázky pre grafiku.

5.2.1 Grafika

Grafika je bezpochyby najjednoduchšou časťou tejto hry. Grafické časti hry sú v rámci tejto bakalárskej práce len:

- Ikony v hlavnom menu – tieto ikony boli získané zo stránky Icons DB[1] alebo vlastnoručne vytvorené.
- Herné pozadia – všetky herné pozadia boli získané zo stránky Pexels[2].
- Fonty – v hre sú použité fonty Molot, SquareFont a Roboto. Všetky fonty boli získané zo stránky DaFont[3], kde sú uvedené ako 100% zadarmo.

Okrem týchto častí hra obsahuje obrázky užívateľského rozhrania, ktoré boli navrhnuté Marekom Farkašom[4] a vytvorené autorom práce.

Ostatné obrázky pre hru boli navrhnuté a vytvorené autorom práce a patria medzi ne východzie výzory blokov a iné obrázky, použité v užívateľskom rozhraní hry.

Výzory blokov je možné užívateľsky meniť, čo je vysvetlené v prílohe, v sekcii A.4. O načítanie užívateľských výzorov pre bloky sa stará statická trieda `SkinLoader` a funguje nasledovne:

1. Hráč musí vytvoriť potrebné zložky v perzistentnom hernom úložisku, kam je treba umiestniť konfiguračný súbor a súbory s obrázkami pre výzor.
2. Po vytvorení týchto zložiek je možné z jednej z nich načítať konfiguráciu výzorov a následne podľa konfigurácie výzorov načítať všetky v nej špecifikované výzory.

   Načítanie výzorov funguje tak, že sa najprv načítajú všetky potrebné súbory, ktoré sú špecifikované v konfigurácii. Následne sa podľa konfigurácie zo súborov vyrežú jednotlivé obrázky.

   Funkcionalita pre načítanie textúr zo súborov a následné vyrezávanie obrázkov je poskytovaná herným engine Unity.
3. Po načítaní všetkých výzorov alebo po neúspechu hra ukáže upozornenie.
4. Ak sa výzor úspešne načítal, objekty blokov v hre pri inicializácii zistia, či pre ne existuje v zozname načítaných výzorov zápis o vlastnom výzore. Ak tento zápis existuje, tak z neho získajú konfiguráciu výzoru aj s potrebnými obrázkami a vytvoria objekty pre zobrazenie výzoru.

Takisto je možné meniť herné pozadia. Toto je tiež vysvetlené v prílohe, v sekcii A.3. O načítanie pozadí sa stará statická trieda `BackgroundPackLoader` a funguje podobne ako `SkinLoader`.

[1] https://www.iconsdb.com/
[2] https://www.pexels.com/
[3] https://www.dafont.com/
[4] http://noobish.eu/

5.2.2 Zvukové efekty

Zvukové efekty pre túto bakalársku prácu boli vytvorené Radoslavom Žilkom. Použité sú len počas samotnej hry a prehrávajú sa pomocou Game Objectu `SoundEffectsPlayer` v hernej scéne.

Systém zvukových efektov funguje pomocou herných udalostí – pri inicializácii hry si objekt `SoundEffectsPlayer` zaregistruje obslužné funkcie pre všetky potrebné udalosti a následne pri nastaní jednotlivých udalostí prehráva zvukové efekty podľa správ, ktoré sú pri udalostiach posielané. Implementácia herných udalostí je vysvetlená v sekcii 5.5.1.

Pokiaľ hráč používa vlastný skript pre prehrávanie zvukových efektov, tak sa vždy hra pokúsi najprv spustiť skript a zaregistrovať funkcie preň. V opačnom prípade hra zaregistruje východzie funkcie. O samotné prehrávanie zvukových efektov sa stará natívny Unity komponent `AudioSource`, ktorý je pripojený ku Game Objectu `SoundEffectsPlayer`.

Zvukové efekty je vďaka tomu možné prehrávať dvomi spôsobmi – buď sa na `AudioSource` zavolá metóda `PlayOneShot()`, čo umožňuje prehrávať niekoľko rôznych klipov cez seba, alebo sa danému `AudioSource` nastaví jeho klip a zavolá sa metóda `Play()`. Pomocou funkcia `Play()` môže byť prehrávaný vždy len jeden zvukový klip, čo môže byť užitočné pre dlhšie zvukové klipy, ktoré sú používané napríklad ako hlášky hlásateľa.

5.2.3 Hudba

Hudba pre hru bola zakúpená na stránke Humble Bundle[5].

O prehrávanie hudby sa stará Game Object `MusicPlayer`, ktorý funguje ako singleton – zostáva aktívny počas celého fungovania hry. Hudbu prehráva podľa typu aktívnej scény. Typ scény je identifikovaný prefixom názvu scény – existujú len dva typy, menu scény a herná scéna. V hernej scéne sa hudba automaticky neprehráva ale miesto toho reaguje na stav hry – začne hrať vždy keď skončí počiatočný odpočet hry a prestane hrať, keď hráč prehrá alebo ukončí hru. S následným reštartom herného módu sa hudba znova spúšťa po skončení odpočtu.

Herná hudba je ďalej rozdelená na skupiny podľa typu hudby. V herných nastaveniach si hráč môže vybrať, akú skupinu hudby chce zrovna prehrávať. Keď sa potom v hre bude vyberať hudba na prehranie, vyberie sa podľa nastavenej skupiny.

Hudbu a zvukové efekty je takisto možné užívateľsky meniť. Toto je podrobne vysvetlené v prílohe, v sekcii A.5. O načítanie vlastnej hudby a efektov sa stará statická trieda `SoundPackLoader`, ktorá načíta súbory zo špecifikovanej zložky. Medzi súbormi pre hudbu je tiež dôležitý súbor hudobnej konfigurácie, v ktorom užívateľ môže špecifikovať, ktorá hudba má byť používaná v menu scénach a aké skupiny hernej hudby má hra ukazovať.

5.3 Systémy nastavení

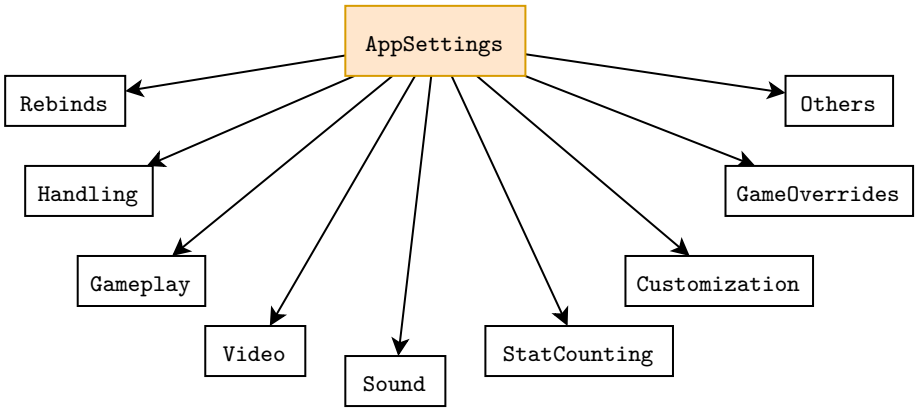
Na systémoch nastavení stojí celá funkcionalita hry. V tejto sekcii bude rozobrané, ako tieto systémy fungujú, ako k nim pristupuje zvyšok hry, a takisto aj ako sú previazané s užívateľským rozhraním.

[5] https://www.humblebundle.com/

5.3.1 Globálne nastavenia

Pre globálne nastavenia v hernom engine Unity existuje systém `PlayerPrefs`, ktorý podľa platformy ukladá nastavenia do rôznych typov perzistentného úložiska. Kvôli väčšej prístupnosti priemernému užívateľovi bol však pre túto bakalársku prácu použitý iný prístup – globálne nastavenia sú uložené v statickej triede `AppSettings`, ku ktorej je možné pristupovať z akejkoľvek časti hry. Pri vypnutí hernej aplikácie sa automaticky uložia na disk do súboru `appSettings.json` v perzistentnom úložisku. Pri opätovnom zapnutí hernej aplikácie sa z daného súboru načítajú a hra sa postará o to, aby sa prispôsobila všetkým nastaveniam.

Pre serializáciu a deserializáciu herných nastavení je použitá knižnica Newtonsoft.Json[6], ktorá je dostupná pod licenciou MIT.



Obr. 5.2: Obrázok znázorňuje rozdelenie aplikačných nastavení na niekoľko rôznych skupín. Tieto skupiny existujú aj v kóde a rozdeľujú nastavenia na časti, v ktorých spolu jednotlivé nastavenia súvisia.

Rozdelenie globálnych nastavení je znázornené na obrázku 5.2. Jednotlivé časti nastavení sú:

- `Rebinds` – jedno nastavenie, v ktorom sú uložené všetky užívateľské priradenia tlačidiel k akciám.
- `Handling` – nastavenia, ktoré určujú detaily ovládania. Sem patria napríklad soft drop faktor, DAS a ARR intervaly a iné.
- `Gameplay` – nastavenia, ktoré nejako prispôsobujú časti samotnej hry, ale nijako nevplývajú na pravidlá. Patria sem napríklad rôzne viditeľnosti častí hry alebo možnosť hru automaticky pozastaviť, keď aplikácia stratí focus.
- `Video` – nastavenia, ktoré sa týkajú vykresľovania hry, napríklad preferovaná obnovovacia rýchlosť, priehľadnosť pozadí a iné.
- `Sound` – nastavenia zvuku. Tieto ovládajú rôzne hlasitosti hudby a zvukových efektov.
- `StatCounting` – nastavenia počítadiel štatistík. Tu sú uložené užívateľsky definované skupiny živých počítadiel štatistík.

[6] https://www.newtonsoft.com/json

- `Customization` – nastavenia zložiek, z ktorých sa majú načítať užívateľské výzory blokov, užívateľské pozadia a užívateľské zvuky.
- `GameOverrides` – v tejto časti nastavení sú nastavenia herných módov, ktoré sú určené na prepísanie východzích nastavení. Pre každý typ hry sa uloží jedna skupina týchto „overrides“.
- `Others` – nastavenia, ktoré s ostatnými nemajú nič spoločné. V tejto bakalárskej práci táto skupina obsahuje len nastavenie zobrazovania aktivity v aplikácii Discord.

Všetky nastavenia, ktoré sú nejakým spôsobom obmedzené na určité hodnoty, sú implementované ako C# properties – pri nastavení sa zistí, či je hodnota valídna a ak nie, tak sa zmení na najbližšiu valídnu hodnotu pre dané nastavenie.

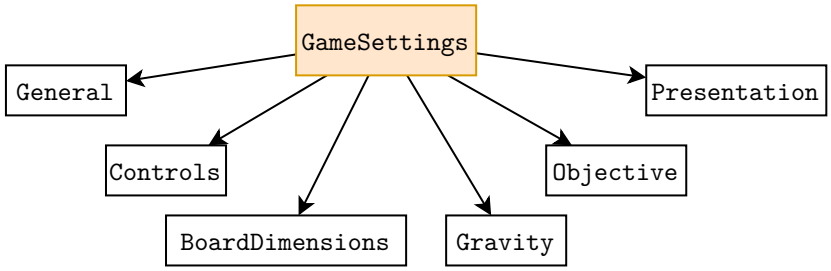
Každé nastavenie, na ktoré je treba nejakým spôsobom reagovať, má implementovaný takzvaný `Applier` Component. Tieto Componenty sú naprogramované tak, aby reagovali na zmenu nastavenia – ak sa ich nastavenie zmení, tak zareagujú a upravia svoju časť aplikácie tak, aby súhlasila s novou hodnotou nastavenia.

Pre príklad, pri zmene zložky s užívateľskými pozadiami sa zistí, či je nová cesta valídna a ak je možné z nej načítať pozadia, pozadia sa načítajú. Po načítaní pozadí sa vyvolá udalosť, na ktorú práve načítané pozadie zareaguje a zmení sa.

Ďalšie zaujímavé príklady `Applier` akcií patrí napríklad vypínanie zvuku, keď aplikácia stratí focus. Všetky nastavenia hlasitostí sú nastavované pomocou Unity AudioMixer objektu. Ak sa nastaví vypínanie zvuku, tak sa na udalosť `Application.focusChanged` zaregistruje obslužná funkcia, ktorá nastaví hlavnú hlasitosť hry.

5.3.2 Herné nastavenia

Herné nastavenia majú podobnú štruktúru ako globálne nastavenia, avšak miesto jednej statickej triedy sú reprezentované Unity Scriptable Objects. Každý Scriptable Object obsahuje všetky potrebné nastavenia pre funkcionalitu hry, ktorú reprezentuje.



Obr. 5.3: Obrázok znázorňuje rozdelenie herných nastavení na niekoľko rôznych skupín. Tieto skupiny existujú aj v kóde a rozdeľujú herné nastavenia na časti, v ktorých spolu jednotlivé nastavenia súvisia.

Rozdelenie herných nastavení je zobrazené na obrázku 5.3. Jednotlivé časti týchto nastavení sú:

- `General` – všeobecné nastavenia, ktoré sa týkajú celej hry. Sú tu napríklad seed hry, typ náhodného generátora dielikov, počet ukážok dielikov a iné.

- `Controls` – nastavenia, ktoré sa týkajú ovládania hry. Obsiahnuté nastavenia sú napríklad možnosti povoliť alebo zakázať rôzne rozširujúce mechaniky, možnosť vynútiť v hernom móde nastavenia zo skupiny `Handling` a možnosť vybrať rotačný systém.
- `BoardDimensions` – tieto nastavenia ovládajú šírku a výšku hernej plochy a rôzne iné výšky, ktoré sú pre hry dôležité.
- `Gravity` – nastavenia v tejto skupine ovládajú gravitáciu, lock delay a iné veci, ktoré sa týkajú pokladania dielikov.
- `Objective` – tieto nastavenia ovládajú skórovanie, úrovne a generovanie odpadových riadkov.
- `Presentation` – nastavenia v tejto skupine ovládajú prezentačnú časť hry, ako odpočet, názov hry a skupinu živých počítadiel štatistík.

U herných nastavení sú podobne ako u globálnych nastavení použité C# properties na validáciu hodnôt nastavení.

U herných nastavení je výhodou, že počas behu hry ich nie je možné meniť, preto nie je treba žiadne Componenty podobné `Applierom` u globálnych nastavení. Niekoľko rôznych nastavení však pri svojej zmene načíta dáta zo súboru – napríklad pri nastavení vlastného generátoru odpadových riadkov sa vždy skript daného generátora načíta priamo do nastavení. Táto funkcionalita je tiež poskytovaná pomocou C# properties.

5.4 Užívateľské rozhranie

V tejto sekcii bude rozobraná funkcionalita užívateľského rozhrania a jeho prepojenie so systémami nastavení. Výzor užívateľského rozhrania pre túto bakalársku prácu bol navrhnutý Marekom Farkašom[7]. Výzor časti užívateľského rozhrania v hre je zobrazený na obrázku 5.4.

5.4.1 Užívateľské rozhranie nastavení

Užívateľské rozhranie nastavení bolo navrhnuté tak, aby bolo možné pri pridávaní jednotlivých nastavení jednoducho vytvoriť aj časti užívateľského rozhrania k nim. Toto je zaistené pomocou niekoľkých vecí:

- Generická abstraktná trieda, ktorá slúži ako základ pre akýkoľvek menič nastavení – táto trieda poskytuje funkcionalitu, ktorá je potrebná pre zmenu akéhokoľvek nastavenia a následné zobrazenie novej hodnoty v UI.
- Prístup k nastaveniam pomocou reflexie – aby nebolo treba robiť pre každé nastavenie vlastnú triedu, je možné nastavenie zmeniť pomocou reflexívnych metód, ktoré použijú názvy properties k tomu, aby sa dostali k správnemu nastaveniu.

  Vďaka tomuto je pre zmenu rôznych nastavení toho istého typu treba napísať len jednu triedu a zmeniť u nej cestu k použitému nastaveniu.
- Používanie `Applier` Componentov z herných nastavení – zmena nastavení je oddelená od ich aplikovania, čo umožňuje pri pridaní nových nastavení jednoducho len napísať nový `Applier`.

[7] „http://noobish.eu/“

Obr. 5.4: Obrázok ukazuje užívateľské rozhranie hlavného menu a ukazuje časť globálnych nastavení aplikácie.

Vďaka tomuto je pre všetky jednoduché typy – čísla, textové reťazce a boolovské hodnoty – možné napísať len jednu triedu pre typ, v ktorej sa rieši len logika špecifická pre daný typ. Aplikovanie nastavení je od tohoto kompletne oddelené a treba ho písať len pre nastavenia, ktoré túto funkcionalitu potrebujú.

Štruktúra tried, ktoré sú použité pre užívateľské rozhranie slúžiace k zmene herných nastavení, je zobrazená na obrázku 5.5.

Problematické je však nastavovanie enum hodnôt, súborov a zložitejších objektov. Pre zložitejšie objekty je treba písať špecializované triedy pre menenie hodnôt, ale pre enumy a súbory existujú špeciálne generické triedy, ktoré poskytujú potrebnú funkcionalitu.

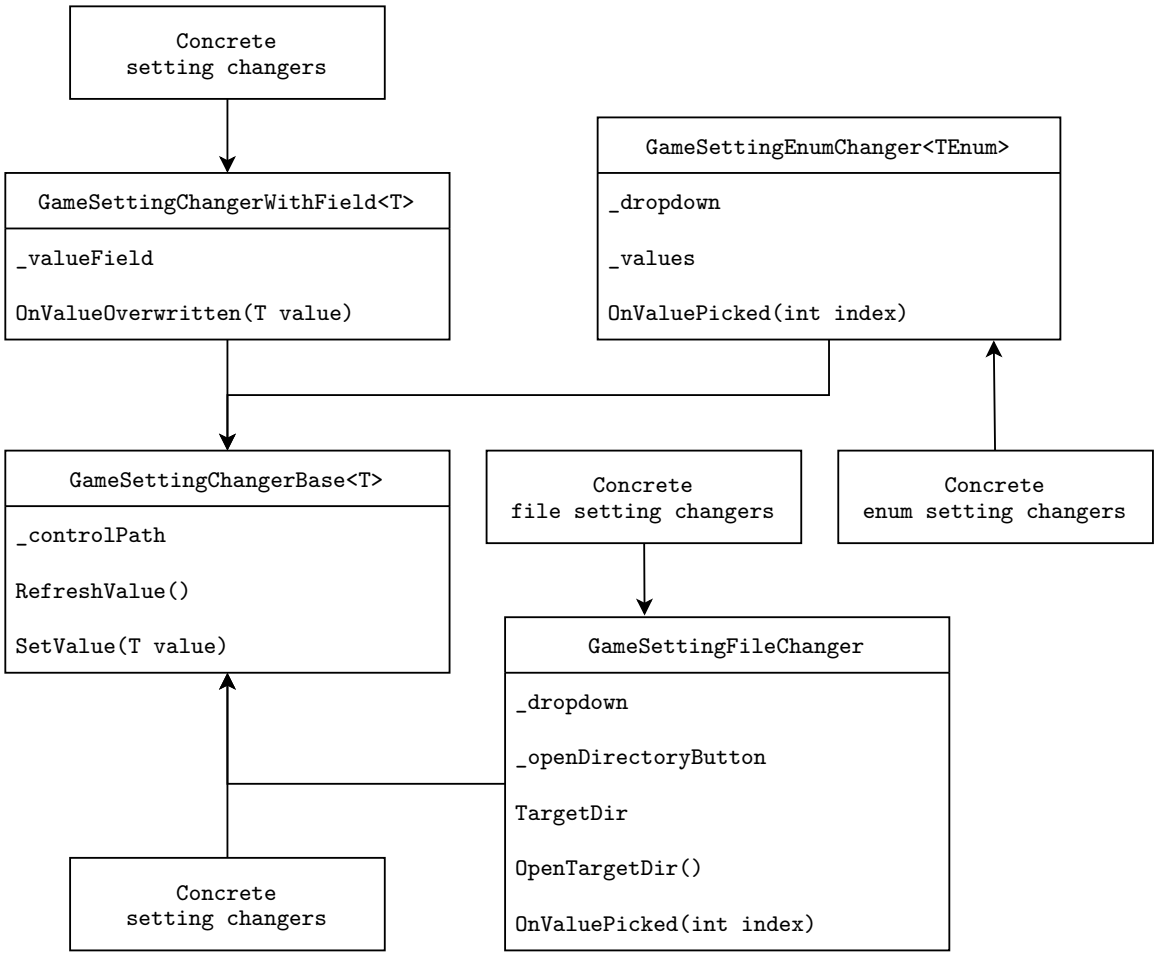
U enumov je použité zjednodušenie týchto hodnôt na číslo – keďže všetky enumy v jazyku C# sú vnútorne reprezentované číselne, je jednoduché ich konvertovať na číslo a v konfigurácii Game Objectu umožniť užívateľovi v editore napísať hodnoty, ktoré má menič enum nastavenia podporovať a názvy, ktoré má pre jednotlivé hodnoty používať.

U súborov je ideálne poskytnúť užívateľovi možnosť vybrať si z ponuky podľa toho, v akej zložke sa použiteľné súbory nachádzajú. Preto existuje trieda `GameSettingFileChanger`, ktorá slúži ako základ pre ostatné triedy pre menenie súborov. Triedy, čo z nej dedia následne musia špecifikovať len cestu k zložke, z ktorej je možné vyberať súbor pre dané nastavenie.

Podobný systém ako v herných nastaveniach je použitý aj pre systém globálnych nastavení.

5.4.2 Ostatné užívateľské rozhranie

Okrem užívateľského rozhrania globálnych a herných nastavení hra takisto obsahuje rozhranie v hre, ktoré je rozdelené na niekoľko častí. Tieto časti sa zobrazujú podľa momentálneho stavu hry, ktorý je známy podľa udalostí (herné udalosti sú vysvetlené ďalej v sekcii 5.5.1).



Obr. 5.5: Obrázok znázorňuje štruktúru tried použitých pre menenie herných nastavení. Základná generická trieda `GameSettingChangerBase<T>` poskytuje všeobecnú funkcionalitu a ďalej z nej dedia trieda pre nastavenia s textovým poľom, trieda pre menenie enum nastavení a trieda pre menenie súborových nastavení. Z nasledujúcich tried je potom možné vytvárať konkrétne triedy, ktoré sa používajú ako komponenty v hernom engine Unity

V hre sú takisto zobrazované ako užívateľské rozhranie počítadlá štatistík – tieto počítadlá sú len jednoduché textové Game Objecty, ktoré dokážu meniť svoj text, farbu a viditeľnosť pomocou užívateľského skriptovania. Užívateľ je taktiež počas hry schopný tieto počítadlá počas hry presúvať a meniť ich veľkosť. Presúvanie a menenie veľkosti je možné pomocou držania a ťahania ľavého tlačidla myši, keď je kurzor umiestnený nad držadlami, ktoré sú pre tieto akcie určené. Následná zmena veľkosti alebo posun je potom vypočítaný pomocou zapísania pôvodnej polohy kurzoru a vypočítania rozdielu od polohy kurzoru v momente pustenia.

5.5 Funkcionalita hry

V tejto sekcii budú rozobrané jednotlivé časti funkcionality hry. Hra sa skladá z mnohých navzájom komunikujúcich častí. Pre správne fungovanie hry je treba aby medzi sebou navzájom komunikovali. Ešte pred spustením hry je treba jednotlivé časti inicializovať tak,

aby odpovedali herným nastaveniam. Po inicializácii hry je následne najdôležitejšie precízne spracovanie užívateľských vstupov a herná logika, ktorú je možné identicky prehrať po uložení herných nastavení a časov vstupov.

5.5.1 Komunikácia pomocou udalostí

Ako hlavný nástroj pre komunikáciu hry slúži objekt `Mediator`. Tento objekt slúži na registrovanie obslužných funkcií pre rôzne typy udalostí a odosielanie príslušných správ, keď dané udalosti nastanú. Keďže tento objekt je úplným základom pre väčšinu komunikácie v hre, bude v tejto sekcii podrobne rozobraný.

`Mediator` podporuje metódy pre registráciu obslužných funkcií – pre zaregistrovanie je treba uviesť typ udalosti, na ktorú chce funkcia reagovať, samotná funkcia a priorita danej funkcie. `Mediator` uchováva zoznam kolekcií funkcií priradených k typom udalosti, pri ktorých majú sa majú dané funkcie zavolať. Funkcie v kolekcii sa pri nastaní udalosti volajú podľa priority v danej kolekcii. Pri registrovaní novej funkcie sa podľa typu udalosti pridá daná funkcia do príslušnej kolekcie.

Takisto je možné zrušiť registráciu funkcií. V tomto prípade sa jednoducho z príslušnej kolekcie použitá funkcia odoberie.

Udalosť sa vyvoláva tak, že na `Mediatore` sa zavolá metóda na poslanie správy – typ správy určuje typ nastanej udalosti a všetky funkcie, ktoré boli pre danú udalosť zaregistrované, sa v poradí podľa priority vyvolajú s poslanou správou ako argumentom.

Existuje niekoľko rôznych typov správ pre všetky dôležité udalosti v hre – správy sa odosielajú pri zmene stavu hry, pri pohybe aktívneho dielika, pri stlačení tlačidla, atď. Všetky herné Componenty si typicky pri začatí hry zaregistrujú funkcie, ktoré budú potrebné udalosti obsluhovať a následne sa pri ukončení hry `Mediator` úplne vyprázdni.

5.5.2 Inicializácia jednotlivých častí hry

Inicializácia jednotlivých častí hry prebieha hneď po načítaní hernej scény. Herné nastavenia sú pred načítaním scény posunuté pomocou statických polí Componentu `GameInitializer`, ktorý má pri načítaní scény prioritu. Vďaka tomu hneď po načítaní scény a vytvorení Game Objectov rozošle objekt s nastaveniami, pomocou ktorého sa všetky Game Objecty náležite nastavia.

Systém rozosielania nastavení je jednoduchý a ľahko rozšíriteľný o ďalšie nastaviteľné časti hry. Každý Component, ktorý potrebuje pre svoju inicializáciu prístup k herným nastaveniam, implementuje interface `IGameSettingsDependency`. Všetky Game Objecty, ktoré obsahujú tieto Componenty, sú detskými Game Objectami `GameInitializera`.
`GameInitializer` pri svojom načítaní všetky tieto Componenty nájde pomocou funkcie `GetComponentsInChildren<T>()` a následne im pošle nastavenia. Po odoslaní nastavení sa každý Component sám nastaví tak, aby s nimi súhlasil.

Medzi Componenty, ktoré sú týmto spôsobom inicializované špecificky pre herné nastavenia, patria napríklad:

- `Board`, ktorý nastavuje dimenzie a generátor odpadových riadkov.
- `GameCamera`, ktorá nastavuje veľkosť reálnej kamery tak, aby bolo vždy viditeľné celé herné pole.

- `PieceSpawner`, ktorý inicializuje náhodný generátor dielikov a takisto ukážky nasledujúcich dielikov.
- `StatCounterManager`, ktorý inicializuje všetky živé počítadlá štatistík.
- `GameManagerParent`, ktorý vytvorí reálny použitý Game Manager Component.

5.5.3 Správa užívateľských vstupov a herná logika

Správa vstupov a herná logika sú spolu pomerne úzko späté, preto sú v spoločnej sekcii. Logika hry je zložená a mnohých častí a veľa z nich je priamo ovplyvňovaná užívateľskými vstupmi.

Skripty, ktoré zaisťujú funkcionalitu hry, sú:

- `Board` – Component slúži pre správu položených dielikov a ich súčastí. Keď užívateľ položí dielik, `Board` sa náležite upraví, zistí, či treba vyčistiť nejaké riadky a prípadne vygeneruje odpadové riadky, ak je to podľa herných nastavení treba.

  Tento Component takisto spravuje životný cyklus vrstiev odpadových riadkov a ich blokov.
- `GhostPiece` – Component zaisťuje funkcionalitu tieňového dielika.
- `WarningPiece` – Component zaisťuje funkcionalitu varovného dielika. Pokiaľ sú bloky na `Boarde` umiestnené nad určitou výškou, začne ukazovať, v akej polohe sa objaví nasledujúci dielik.
- `PieceContainer` – tento Component slúži pre uchovanie dielikov v určitej polohe. Je používaný pre funkcionalitu mechaniky hold a takisto pre ukážky nasledujúcich dielikov.
- `EndConditionChecker` – Component sa stará o zisťovanie, či bola naplnená podmienka pre ukončenie hry a pri jej splnení vyvolá úspešný koniec.
- `GameManager` – tento Component slúži pre správu úrovní hry a počítanie skóre.
- `GameStateManager` – Component je používaný na správu stavu hry. Pokiaľ nejaký iný Component potrebuje hru pozastaviť, ukončiť, alebo reštartovať, volá metódy na tomto Componente. Všetky správy o stave hry sa posielajú výhradne z tohoto Componentu.
- `PieceSpawner` – Component slúži pre vytváranie dielikov, ich ukazovanie v ukážkach a ich objavenie na správnej časť hernej plochy.

  Takisto spravuje životný cyklus objektov dielikov.
- `InputProcessor` – tento Component sa stará o správu veškerých užívateľských vstupov, ktoré sa týkajú hry. Bude podrobnejšie rozobraný ďalej v tejto sekcii.
- `Timer` – Component sa stará o presné časovanie hry a o menenie času v prehrávkach tak, aby stav hry vždy zodpovedal nastavenému času.
- `StatCounterManager` – Component spravuje živé počítadlá štatistík a počíta všetky vstavané štatistiky v hre okrem skóre.

- `Recorder` – Component sa stará o zaznamenávanie užívateľských vstupov do prehrávok a ich ukladanie do súborov.
- `CrashHandler` – Component sa stará o spracovanie chybových stavov hry, ktoré môžu nastať vďaka užívateľskému skriptovaniu. Pokiaľ nastane chyba, po ktorej nie je možné pokračovať v hre (napríklad generátor dielikov vygeneruje typ dielika, ktorý nie je hrou podporovaný), tak tento Component zobrazí hlásenie o chybe a poskytne hráčovi možnosť vrátiť sa do hlavného menu.
- `SeedSetter` – Component vždy pri štarte alebo reštarte hry upozorní všetky ostatné časti hry o zmene seedu pre jednotlivé náhodné generátory.
- `SoundEffectsPlayer` – Component sa stará o prehrávanie zvukových efektov.
- `Mediator` – Component slúži pre vyvolávanie herných udalostí a pre registrovanie obslužných funkcií.
- `Piece` – Component je priradený ku každému dieliku a slúži pre zoskupovanie jednotlivých blokov do dielikov tak, aby bolo možné ich používať dookola.
- `PieceBlock` – Component je priradený ku každému bloku a stará sa o to, aby sa pre daný blok správne načítali výzory a aby blok správne reagoval na vyčistenie.

Pre ušetrenie pamäti je pre mnohé časti hry použitý návrhový vzor „Pool“. Tento návrhový vzor funguje tak, že miesto toho, aby sa objekty neustále vytvárali a ničili, sú len aktivované a deaktivované. Životný cyklus objektov, ktoré sú takto spravované, funguje nasledovne:

1. Pri prvom využití objektu sa objekt vytvorí a alokuje sa preň potrebná pamäť.
2. Až daný objekt nie je treba, miesto zničenia a dealokácie sa jednoducho deaktivuje. Toto je rôzne podľa typov objektov.
3. Až je daný typ objektu znovu treba využiť, miesto opätovného vytvorenia sa aktivuje. Toto môže byť znova rôzne podľa typov objektov.
4. Keď sa rozhodne, že daný objekt už nie je treba, alebo ak sa odíde z hry, je daný objekt dealokovaný.

Tento návrhový vzor je použitý na niekoľkých miestach, ale hlavne pre správu životných cyklov dielikov, vrstiev odpadových riadkov a odpadových blokov.

Skôr spomenutý Component `InputProcessor` je zodpovedný za spracovanie užívateľského vstupu. Pomocou Unity Input Systemu reaguje na užívateľské akcie – s každou akciou, teda stlačením alebo pustením tlačidla, sa `InputProcessoru` odošle udalosť. Ak je daná udalosť v herných nastaveniach povolená, tak sa spracuje. Spracovanie akcií funguje nasledovne:

1. Vytvorí sa vnútorná reprezentácia akcie, ktorá obsahuje typ akcie a čas akcie od začiatku hry.
2. Pokiaľ je hra pozastavená, akcia sa zapíše do zoznamu držaných akcií.

3. Pokiaľ hra nie je pozastavená, tak sa podľa typu akcie vykonajú všetky potrebné reakcie – napríklad pokiaľ hráč stlačil tlačidlo pre pohyb doľava, tak sa nastaví čas počiatku držania tohoto tlačidla, prípadne sa nastaví čas aktivácie mechaniky DAS a pokiaľ existuje aktívny dielik a je voľné miesto na hernej ploche, tak sa aktívny dielik pohne doľava.

Okrem reagovania na užívateľský vstup je `InputProcessor` takisto zodpovedný za reagovanie na posun herného času – toto zahŕňa posúvanie dielikov pomocou mechaniky DAS, padanie dielikov, aktivácia mechaniky softdrop, ak je nastavené jej oneskorenie a uzamknutie dielika, pokiaľ kvôli gravitácii dopadol na miesto.

Tieto udalosti majú naplánovaný čas ich nastania – pri obnovení stavu hry v metóde `Update()` `InputProcessor` vždy zoberie udalosť, ktorá je nastavená na najbližší čas, zistí, či tento čas už nastal a ak áno, tak danú udalosť vykoná. Pri vykonaní udalosti sa môže nastaviť jej ďalší čas na nejakú odvodenú hodnotu alebo na pozitívne nekonečno, ak udalosť nemá automaticky nastávať znova. Napríklad padnutie dielika kvôli gravitácii a pohyb mechanikou DAS sa automaticky opakujú, ale aktivácia DAS či objavenie dielika sa musia znova nastaviť iným spôsobom.

Vďaka tomuto systému spracovania najbližších udalostí je pri opätovnom prehraní hry vždy priebeh hry úplne rovnaký ako pri pôvodnom hraní.

Kapitola 6

Záver

Cieľom tejto práce bolo vytvoriť príjemnú blockstacker hru s čo najväčším počtom nastavení a možností, ktoré dokážu hráčsky zážitok vylepšiť. Zároveň bolo zámerom hru vytvoriť tak, aby bola ideálna na použitie v súťažnom prostredí.

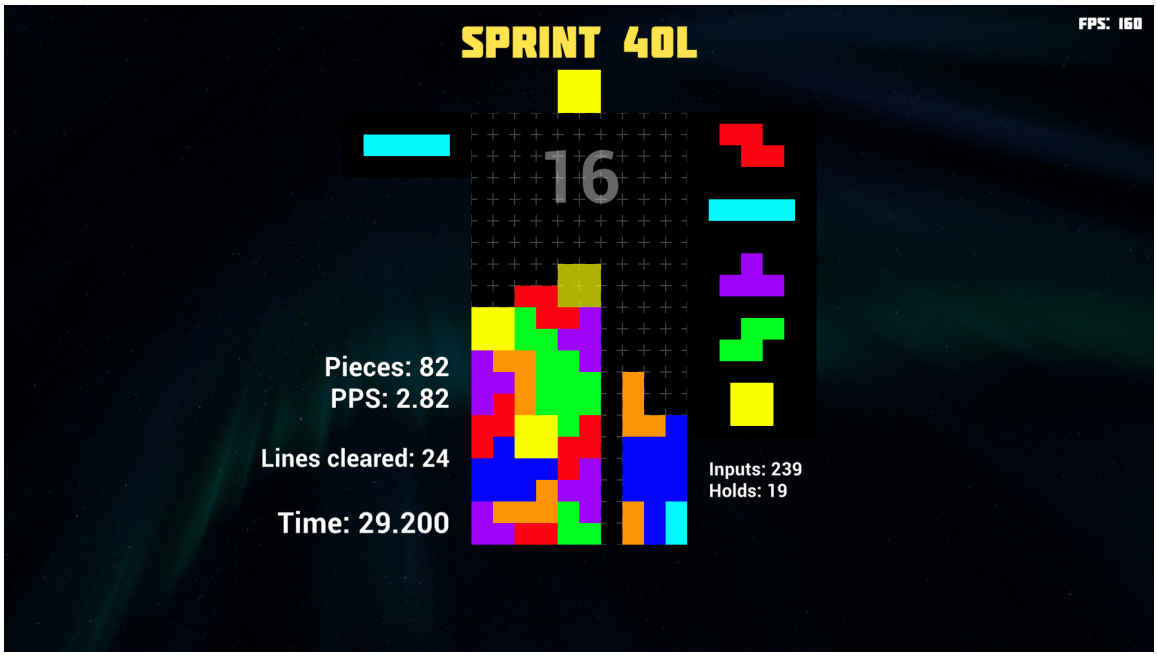
Pre implementáciu hry bolo potrebné naštudovať niekoľko moderných hier rovnakého žánra, ktorými bola bakalárska práca inšpirovaná. Okrem toho bolo treba naštudovať si funkcionalitu herného engine Unity, z ktorej boli najdôležitejšie rôzne spôsoby, akým je v Unity spracovávaný užívateľský vstup a systémy na tvorbu kvalitného užívateľského rozhrania.

Najzložitejším systémom v hre je systém spracovania užívateľského vstupu, ktorý dokáže reagovať na vstupy s väčšou presnosťou ako akékoľvek podobné hry a takisto je viac konfigurovateľný.

Ďalej aplikácia hry podporuje veľkú škálu herných mechaník od štandardných, cez inšpirované až k úplne originálnym. Takisto podporuje ukladanie a prehrávanie prehrávok, veľké množstvo možností prispôsobenia hernej aplikácie a vďaka knižnici NLua schopnosť užívateľsky implementovať živé počítadlá štatistík, náhodné generátory dielikov, generátory odpadových riadkov a dokonca aj systémy, ktoré spravujú skórovanie a úrovne hry.

Zámer práce bol splnený – hra ponúka mnohé možnosti, ktoré v žiadnej inej hre z rovnakého žánra nie sú k dispozícii a vďaka veľmi presnému systému spracovania užívateľského vstupu je vhodná na použitie v súťažnom prostredí. Hra takisto ponúka veľké možnosti na ďalší rozvoj, medzi ktoré patrí implementácia rôznych módov pre viacerých hráčov, zdieľanie prehrávok a nastavení užívateľských módov online a vytvorenie rebríčkov pre všetky módy, či už užívateľské alebo vstavané.

Výzor finálnej hernej scény je zobrazený na obrázku 6.1. Hra je dostupná na stránke https://mrakdun-desu.itch.io/ustacker.



Obr. 6.1: Obrázok zobrazuje výzor finálnej hernej scény. Na obrázku je hra uprostred vstavaného herného módu s názvom Sprint 40L.

Literatúra

[1] *Tetris Wiki* [online]. 2007. Revidované 17. 10. 2020 [cit. 2023-5-10]. Dostupné z: https://tetris.wiki/.

[2] PUC Rio. *Lua Official Website* [online]. 2022 [cit. 2023-5-10]. Dostupné z: https://www.lua.org/.

[3] Technologies, U. *Unity - Manual: Unity User Manual 2022.2* [online]. 2023 [cit. 2023-5-10]. Dostupné z: https://docs.unity3d.com/2022.2/Documentation/Manual/index.html.

Príloha A

Manuál k aplikácii

Hra `UStacker` je navrhnutá k použitiu na 64bitovom operačnom systéme Windows alebo Linux. Perzistentné užívateľské nastavenia sa ukladajú na súborový systém a ich východzia lokácia je `%userprofile%\AppData\LocalLow\Mrak\UStacker` pre systém Windows a `$HOME/.config/unity3d` alebo `$XDG_CONFIG_HOME/unity3d` pre systém Linux.

Táto dokumentácia slúži hlavne ako na návod pre prispôsobenie výzoru, zvuku a pravidiel hry. Obsahuje všetky informácie potrebné pre písanie skriptov, ktoré hra dokáže prečítať a interpretovať.

A.1 Používanie jazyka Lua v užívateľských skriptoch

Jazyk Lua je pre všetky užívateľské skripty mierne upravený.

Odstránená funkcionalita

Pre bezpečnosť boli odstránené nasledujúce funkcie a objekty:

- `os`
- `io`
- `require`
- `dofile`
- `package`
- `luanet`
- `load`
- `import`

Všetky tieto hodnoty sú v užívateľských skriptoch nastavené na hodnotu `nil`.

Upravená funkcionalita

Funkcia `math.random` bola v užívateľských skriptoch nahradená náhodným generátorom implementovaným v hre. Funkcia funguje rovnako ako natívna funkcia `math.random`.

Keďže je `math.random` nahradená generátorom vstavaným do hry, funkcia `math.randomseed`, ktorá je zvyčajne používaná na nastavenie seedu náhodného generátoru v užívateľských skriptoch hry nemá žiadny efekt. Seed náhodného generátoru je vždy pevne nastavený na začiatku hry a užívateľ ho nemôže zmeniť.

Pridaná funkcionalita

Pre ladenie užívateľských skriptov bola pridaná funkcia `DebugLog`. Táto funkcia prijíma jeden parameter a slúži k vypísaniu daného parametra do súboru `logs.txt` v perzistentnom úložisku aplikácie.

A.2 Herné udalosti

Väčšina užívateľského skriptovania v hre funguje na princípe reakcií na udalosti, ktoré sa v hre dejú. Pri každej takejto udalosti je všetkým skriptom poslaná správa s informáciami, ktoré sa k danej udalosti vzťahujú. V tejto sekcii je vysvetlené, ako v skriptoch zaregistrovať na dané udalosti obslužné funkcie, a sú tu popísané všetky typy udalostí, ktoré v hre môžu nastať.

Registrovanie obslužných funkcií na udalosti

Registrovanie obslužných funkcií funguje vrátením tabuľky, ktorá priraďuje názov udalosti k funkcii, ktorá má byť pri nastaní danej udalosti zavolaná. Príklad zaregistrovania obslužných funkcií k udalostiam je uvedený na výpise A.1. Pri pokuse o zaregistrovanie neplatnej udalosti hra zobrazí varovanie.

```
function HandlePiecePlaced(message)
    -- this function will handle the PiecePlaced event
end

function HandlePieceMoved(message)
    -- this function will handle the PieceMoved event
end

return {
    ["PiecePlaced"] = HandlePiecePlaced,
    ["PieceMoved"] = HandlePieceMoved
}
```

Výpis A.1: Príklad registrovania obslužných funkcií k udalostiam v Lua skripte

Udalosť `CountdownTicked`

Táto udalosť nastáva pri každom tikaní odpočtu hry. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`RemainingTicks`	`uint`	Zostávajúci počet tikaní odpočtu

Udalosť `GameEndConditionChanged`

Táto udalosť nastáva pri zmene podmienky ukončenia hry. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`ConditionName`	`string`	Názov podmienky ukončenia hry, ktorá sa práve zmenila
`CurrentCount`	`double`	Časť podmienky, ktorá už bola naplnená
`TotalCount`	`double`	Celkový počet udalostí, ktoré musia do konca hry nastať
`Time`	`double`	Čas od začiatku hry v sekundách

Udalosť `GameStateChanged`

Táto udalosť nastáva pri každom zmene stavu hry. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`PreviousState`	`GameState`	Stav, v ktorom bola hra do tohoto momentu
`NewState`	`GameState`	Stav, v ktorom je hra od tohto momentu
`IsReplay`	`bool`	Naznačuje, či sa hra momentálne prehráva
`Time`	`double`	Čas od začiatku hry v sekundách

Typ `GameState` je enum typ, takže pre porovnanie v Lua skriptoch je treba na ňom zavolať metódu `:ToString()`. Hodnoty, ktoré tento typ môže nadobudnúť:

- `"Unset"` – hra je v tomto stave keď sa herná scéna prvý raz načíta.
- `"Initializing"` – hra je v tomto stave počas toho ako sa jej komponenty inicializujú.
- `"StartCountdown"` – hra je v tomto stave počas počiatočného odpočtu hry.
- `"Running"` – hra je v tomto stave počas behu samotnej hry.
- `"Paused"` – hra je v tomto stave keď je zrovna pozastavená.
- `"ResumeCountdown"` – hra je v tomto stave počas odpočtu po spustení hry z pozastaveného stavu.
- `"Ended"` – hra je v tomto stave po úspešnom ukončení.
- `"Lost"` – hra je v tomto stave po neúspešnom ukončení.

Udalosť `HoldUsed`

Táto udalosť nastáva pri každom stlačení tlačidla pre hold. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`WasSuccessful`	`bool`	Naznačuje, či bola hold akcia povolená
`Time`	`double`	Čas od začiatku hry v sekundách

Udalosť `InputAction`

Táto udalosť nastáva pri každom stlačení akéhokoľvek ovládacieho tlačidla. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`ActionType`	`ActionType`	Typ akcie, ktorá sa práve stala
`KeyActionType`	`KeyActionType`	Typ akcie, ktorá sa stala s tlačidlom
`Time`	`double`	Čas od začiatku hry v sekundách

Typy `ActionType` a `KeyActionType` sú enum typ, takže pre porovnanie v Lua skriptoch je treba na nich zavolať metódu `:ToString()`. Hodnoty, ktoré môže nadobudnúť typ `ActionType`:

- `"MoveLeft"` – pohyb doľava
- `"MoveRight"` – pohyb doprava
- `"Hold"` – hold akcia
- `"Harddrop"` – harddrop akcia
- `"Softdrop"` – softdrop akcia
- `"RotateCW"` – rotácia po smere hodinových ručičiek
- `"RotateCCW"` – rotácia proti smeru hodinových ručičiek
- `"Rotate180"` – rotácia o 180 stupňov
- `"MoveToLeftWall"` – posun k ľavej stene
- `"MoveToRightWall"` – posun k pravej stene

Hodnoty, ktoré môže nadobudnúť typ `KeyActionType`:

- `"KeyUp"` – tlačidlo prestalo byť stlačené
- `"KeyDown"` – tlačidlo bolo stlačené

Táto udalosť nastáva aj v prípade, že akcia nebola úspešná. Pre reálne pohyby kúskov alebo výsledky akcií je treba používať iné udalosti.

Udalosť `LevelChanged`

Táto udalosť nastáva pri každej zmene úrovne. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`Level`	`string`	Názov novej úrovne
`Time`	`double`	Čas od začiatku hry v sekundách

Udalosť `LevelUpConditionChanged`

Táto udalosť nastáva pri každej zmene podmienky pre zmenu úrovne. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`ConditionName`	`string`	Názov podmienky pre zmenu úrovne
`CurrentCount`	`double`	Časť podmienky, ktorá už bola splnená
`TotalCount`	`double`	Celkový počet alebo množstvo podmienky pre zmenu úrovne
`Time`	`double`	Čas od začiatku hry v sekundách

Udalosť `PieceMoved`

Táto udalosť nastáva pri každom pohybe aktívneho dielika. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`X`	`int`	Počet jednotiek na X osi o koľko sa dielik pohol
`Y`	`int`	Počet jednotiek na Y osi o koľko sa dielik pohol
`WasHardDrop`	`bool`	Naznačuje, či bol na tento pohyb použitý harddrop
`WasSoftDrop`	`bool`	Naznačuje, či bol na tento pohyb použitý softdrop
`HitWall`	`bool`	Naznačuje, či sa dielik dotkol pravej alebo ľavej steny hernej plochy
`Time`	`double`	Čas od začiatku hry v sekundách

Udalosť `PiecePlaced`

Táto udalosť nastáva pri každom položení dielika. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`PieceType`	`string`	Typ položeného dielika
`LinesCleared`	`uint`	Počet riadkov vyčistených týmto položením
`GarbageLinesCleared`	`uint`	Počet odpadových riadkov vyčistených týmto položením
`CurrentCombo`	`uint`	Dĺžka momentálneho combo reťazca
`CurrentBackToBack`	`uint`	Dĺžka momentálneho back-to-back reťazca
`BrokenCombo`	`bool`	Naznačuje, či bol týmto položením prerušený combo reťazec
`BrokenBackToBack`	`bool`	Naznačuje, či bol týmto položením prerušený back-to-back reťazec
`WasAllClear`	`bool`	Naznačuje, či položením nastal all clear
`WasSpin`	`bool`	Naznačuje, či položenie bolo identifikované ako spin
`WasSpinMini`	`bool`	Naznačuje, či položenie bolo identifikované ako spin mini
`WasSpinRaw`	`bool`	Naznačuje, či by položenie bolo identifikované ako spin pri detekcii všetkých spinov
`WasSpinMiniRaw`	`bool`	Naznačuje, či by položenie bolo identifikovanéako spin mini pri detekcii všetkých spinov
`WasBtbClear`	`bool`	Naznačuje, či položenie bolo identifikované ako back-to-back vyčistenie
`TotalRotation`	`int`	Celková rotácia vykonaná s dielikom od objavenia v stupňoch proti smeru hodinových ručičiek
`TotalMovement`	`Vector2Int`	Celkový pohyb vykonaný s dielikom od objavenia
`BlockPositions`	`Vector2Int[]`	Pozície položených blokov dielika
`Time`	`double`	Čas od začiatku hry v sekundách

Typ `Vector2Int` je vstavaný typ herného engine Unity. Obsahuje polia `x`, `y`, `magnitude` a `sqrMagnitude`. Dokumentácia k nemu je dostupná na tomto URL: https://docs.unity3d.com/2022.2/Documentation/ScriptReference/Vector2Int.html.

Udalosť `PieceRotated`

Táto udalosť nastáva pri každej rotácii dielika. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`PieceType`	`string`	Typ rotovaného dielika
`StartRotation`	`RotationState`	Rotačný stav dielika pred rotáciou
`EndRotation`	`RotationState`	Rotačný stav dielika po rotácii
`WasSpin`	`bool`	Naznačuje, či rotácia bola identifikovaná ako spin
`WasSpinMini`	`bool`	Naznačuje, či bola rotácia identifikovaná ako spin mini
`WasSpinRaw`	`bool`	Naznačuje, či by položenie bolo identifikované ako spin pri detekcii všetkých spinov
`WasSpinMiniRaw`	`bool`	Naznačuje, či by položenie bolo identifikované ako spin mini pri detekcii všetkých spinov
`Time`	`double`	Čas od začiatku hry v sekundách

Typ `RotationState` je enum typ, takže pre porovnanie v Lua skriptoch je treba na ňom zavolať metódu `:ToString()`. Hodnoty, ktoré tento typ môže nadobudnúť:

- `"Zero"` – východzí rotačný stav.
- `"One"` – dielik je otočený o 90 stupňov po smere hodinových ručičiek.
- `"Two"` – dielik je otočený o 180 stupňov.
- `"Three"` – dielik je otočený o 90 stupňov proti smeru hodinových ručičiek.

Udalosť `PieceSpawned`

Táto udalosť nastáva pri každom objavení dielika. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`SpawnedPiece`	`string`	Typ práve objaveného dielika
`NextPiece`	`string`	Typ nasledujúceho dielika
`Time`	`double`	Čas od začiatku hry v sekundách

Udalosť `ScoreChanged`

Táto udalosť nastáva pri každej zmene skóre. Správa, ktorá je pri udalosti odoslaná, obsahuje nasledujúce informácie:

Názov poľa	Typ poľa	Popis
`Score`	`long`	Nová hodnota skóre
`Time`	`double`	Čas od začiatku hry v sekundách

A.3 Používanie vlastných herných pozadí

Pre použitie vlastných pozadí je treba v perzistentnom úložisku aplikácie vytvoriť zložku `styleCustomization/backgroundPacks`. Túto zložku je takisto možné vytvoriť a otvoriť z herných nastavení.

Pre vytvorenie a použitie balíčku pozadí je treba nasledovať tieto kroky:

1. Vymyslieť názov balíčku a v zložke pozadí vytvoriť zložku s daným názvom.

2. Vložiť súbory pozadí do vytvorenej zložky a správne ich pomenovať.

3. Vybrať v nastaveniach aplikácie nový vytvorený balíček pozadí.

Podporované názvy súborov pozadí sú:

Časť aplikácie	Názov súboru alebo zložky
Všetky časti aplikácie	`default`
Hlavné menu	`mainMenu`
Herné nastavenia	`gameSettings`
Scéna hry	`game`

Ako pozadie je možné používať jeden súbor vo formáte `png` alebo `jpg`. Prípona súboru do názvu pozadia nie je počítaná.

Takisto je možné používať zložku súborov vo formáte `png` alebo `jpg`. V tomto prípade sa pri načítaní danej scény náhodne vyberie jeden z obsiahnutých obrázkov.

Pre príklad, ak užívateľ chce vo všetkých scénach používať jeden z obrázkov 1, 2 a 3, ale v scéne hry chce používať jedine obrázok 4, tak je treba obrázky 1 2 a 3 vložiť do zložky s názvom `default` (názvy obrázkov v zložke môžu byť ľubovoľné) a obrázok 4 pomenovať `game.png`.

A.4 Používanie vlastných výzorov pre bloky

Pre použitie vlastných výzorov pre bloky je treba v perzistentnom úložisku aplikácie vytvoriť zložku `styleCustomization/skins`. Túto zložku je takisto možné vytvoriť a otvoriť z herných nastavení.

Pre vytvorenie a použitie vlastných výzorov pre bloky je treba nasledovať tieto kroky:

1. Vymyslieť názov pre výzor a v zložke výzorov vytvoriť zložku s daným názvom.

2. Vytvoriť súbory pre výzor. Podporované formáty sú `png` a `jpg`.

3. Vytvoriť súbor s konfiguráciou výzoru s názvom `skinConfig.json`.

4. Vložiť súbory pre výzor a súbor konfigurácie do vytvorenej zložky výzoru.

5. Vybrať v nastaveniach aplikácie nový vytvorený výzor pre bloky.

Súbor `skinConfig.json`

Tento súbor obsahuje všetky potrebné informácie pre načítanie a používanie výzoru, ktorý popisuje. Je v ňom možné špecifikovať z ktorých súborov sa obrázky pre výzory majú načítať a ako má výzor fungovať.

Tento súbor obsahuje zoznam záznamov o jednotlivých výzoroch. Pre každý typ výzoru pre blok je treba špecifikovať jeden záznam. Príklad takéhoto záznamu je zobrazený na výpise A.2.

```
{
    "SkinType": "i",
    "BlockNumbers": [0, 1, 2, 3],
    "Layer": 0,
```

```
    "IsConnected": false,
    "RotateWithPiece": false,
    "AnimationFps": 60.0,
    "Sprites": [
        {
            "Filename": "skin.png",
            "LoadFromUrl": false,
            "PixelsPerUnit": 30.0,
            "PivotPoint": {
                "x": 0.5,
                "y": 0.5
            },
            "SpriteStart": {
                "x": 186.0,
                "y": 0.0
            },
            "SpriteSize": {
                "x": 30.0,
                "y": 30.0
            }
        }
    ],
    "ConnectedSprites": {
        "7": []
    }
}
```

Výpis A.2: Výpis znázorňuje príklad záznamu o výzore blokov dielika I.

- `SkinType` – typ bloku, na ktorý bude daný výzor aplikovaný. Valídne hodnoty sú:
  - `"i"` – bloky dielika I
  - `"j"` – bloky dielika J
  - `"l"` – bloky dielika L
  - `"o"` – bloky dielika O
  - `"s"` – bloky dielika S
  - `"t"` – bloky dielika T
  - `"z"` – bloky dielika Z
  - `"usedHold"` – bloky odloženého dielika keď nie je mechanika hold prístupná
  - `"ghost"` – bloky tieňového dielika
  - `"warning"` – bloky varovného dielika
  - `"grid"` – bloky hernej mriežky
  - `"garbage"` – vygenerované odpadové riadky
- `BlockNumbers` – zoznam čísiel blokov, na ktoré bude daný výzor aplikovaný. Toto funguje len pre bloky dielikov, ktoré majú štandardné štyri bloky.

- `Layer` – v prípade, že pre daný blok je špecifikovaných niekoľko výzorov je toto úroveň na ktorej bude obrázok tohto výzoru vykresľovaný. Čím vyššie je číslo, tým vyššia bude priorita vykresľovania.
- `IsConnected` – pokiaľ bude toto pole obsahovať hodnotu `true`, tak sa obrázky budú načítať podľa poľa `ConnectedSprites`, miesto poľa `Sprites`.
- `RotateWithPiece` – pokiaľ bude toto pole obsahovať hodnotu `true`, tak sa pri otáčaní aktívneho dielika bude spolu s ním otáčať aj tento výzor.
- `AnimationFps` – pokiaľ je tento výzor animovaný, tak táto hodnota určite počet snímkov za sekundu, na ktoré bude animovaný. Môže obsahovať desatinné hodnoty.
- `Sprites` – zoznam záznamov o obrázkoch, ktoré sa majú v rámci tohoto výzoru používať. Pokiaľ tento výzor nemá byť animovaný, tak by tento zoznam mal obsahovať len jeden záznam o obrázku.
  - `Filename` – cesta k súboru, v ktorom sa tento obrázok nachádza. Pokiaľ je `LoadFromUrl` nastavené na `false`, toto je brané ako cesta relatívna k zložke s konfiguračným súborom. V opačnom prípade sa toto berie ako URL.
  - `LoadFromUrl` – pokiaľ bude toto pole obsahovať hodnotu `true`, tento obrázok sa bude načítavať z internetu. Toto načítavanie sa deje pri každom štarte aplikácie.
  - `PixelsPerUnit` – počet pixelov na jeden blok.
  - `PivotPoint` – relatívna normalizovaná pozícia v obrázku, ktorá bude braná ako stred daného obrázku. Východzia hodnota je `{"x": 0.5, "y": 0.5}`.
  - `SpriteStart` – pozícia daného obrázku v súbore v pixeloch. Pixely sú počítané od ľavého spodného rohu obrázku. Východzia hodnota je `{"x": 0, "y": 0}`.
  - `SpriteSize` – veľkosť daného obrázku v pixeloch. Rastie smerom doprava a dohora. Východzia hodnota je `{"x": 64, "y": 64}`.
- `ConnectedSprites` – zoznam podporovaných konfigurácií prepojeného výzoru a k nim priradené záznamy o obrázkoch. Konfigurácie sú reprezentované bitovým poľom, v ktorom každý bit naznačuje, či má daná konfigurácia na určitom mieste roh alebo hranu. Polohy jednotlivých bitov:
  - 1 – Vrchná strana.
  - 2 – Ľavá strana.
  - 4 – Pravá strana.
  - 8 – Spodná strana.
  - 16 – Ľavý horný roh.
  - 32 – Pravý horný roh.
  - 64 – Ľavý dolný roh.
  - 128 – Pravý dolný roh.

  Rohové bity by mali byť nastavené len v prípade, že okolo neho nie sú hrany. Pre príklad, ak má konfigurácia hrany na vrchnej a ľavej strane a okrem toho pravý dolný roh, tak by mala byť popísaná číslom 131 (kombinácia ľavej a hornej strany a pravého dolného rohu), nie číslom 147 (kombinácia ľavej a hornej strany, pravého dolného rohu a ľavého horného rohu).

A.5 Používanie vlastného audia

Pre použitie vlastného audia je treba v perzistentnom úložisku aplikácie vytvoriť zložku `styleCustomization/soundPacks`. Túto zložku je takisto možné vytvoriť a otvoriť z herných nastavení.

Vlastné audio je zložené z 2 častí – hudby a zvukových efektov.

Pre vytvorenie a použitie vlastného balíčku audia je treba nasledovať tieto kroky:

1. Vymyslieť názov balíčku a v zložke audia vytvoriť zložku s daným názvom.
2. V prípade používania vlastnej hudby je možné vytvoriť podzložku `music` a vložiť do nej audio súbory pre hudbu. Prehrávanie hudby môže byť ďalej upravené vytvorením súboru `musicConfig.json`.
3. V prípade používania vlastných zvukových efektov je možné vytvoriť podzložku `soundEffects` a vložiť do nej audio súbory pre zvukové efekty. Prehrávanie zvukových efektov môže byť ďalej upravené vytvorením súboru `soundEffects.lua`.
4. Vybrať v nastaveniach aplikácie nový vytvorený výzor pre bloky.

Súbor `musicConfig.json`

Tento súbor umožňuje vybrať si, aká hudba hrá v akej časti hry. Bez tohoto súboru budú všetky audio súbory zo zložky `music` prehrávané len v hernej scéne. Pomocou tejto konfigurácie je možné vybrať si hudbu v menu a vytvoriť skupiny pre hernú hudbu. Z týchto skupín sa dá následne vyberať v herných nastaveniach jednotlivých hier. Príklad konfiguračného súboru je uvedený vo výpise A.3.

```
{
    "MenuMusic": [
        "myMenuMusic1",
        "myMenuMusic2"
    ],
    "GameMusicGroups": {
        "Relax": [
            "relaxingMusic1",
            "relaxingMusic2"
        ],
        "Tryhard": [
            "tryhardMusic1",
            "tryhardMusic2"
        ]
    }
}
```

Výpis A.3: Výpis znázorňuje príklad pre obsah konfiguračného súboru pre hudbu.

Vysvetlenie jednotlivých častí konfigurácie:

- `MenuMusic` – zoznam hudby, ktorá má byť hraná v hlavnom menu a v herných nastaveniach. Pri načítaní menu scény sa náhodne vyberie jeden zo špecifikovaných súborov.

- `GameMusicGroups` – zoznam skupín hudby, ktorá sa má hrať v hre. Názvy jednotlivých skupín môžu byť ľubovoľné, ale nemôžu sa opakovať. V herných nastaveniach sa bude z týchto skupín dať vybrať skupina, ktorá bude hrať v hre.

Obsahom všetkých zoznamov v tomto súbore musia byť názvy súborov v zložke hudby bez prípony. Podporované audio formáty sú:

- `mp3`
- `ogg`
- `wav`
- `aif`
- `mod`
- `s3m`
- `xm`

Úprava zvukových efektov

Bez skriptu `soundEffects.lua` je možné len nahradiť používané zvukové efekty v hre za iné. Tieto efekty sa dajú nahradiť pomocou pomenovania súborov tak, aby názov súhlasil s názvom nejakého z podporovaných efektov. Podporované efekty sú:

Názov súboru	Udalosť, pri ktorej sa klip prehráva
spin	Keď hráč vykoná spin
rotate	Keď hráč vykoná rotáciu bez spinu
move	Keď sa aktívny dielik pohne doprava alebo doľava
softdrop	Keď sa aktívny dielik pohne dole a softdrop je aktívny
hold	Keď sa úspešne použije mechanika hold
allclear	Keď hráč vykoná all-clear
floor	Keď sa dielik položí
clearspin	Keď sa dielik položí pomocou spinu a riadky sú vyčistené
combobreak	Keď hráč preruší combo reťazec
btb_break	Keď hráč preruší back-to-back reťazec
clearline	Keď hráč vyčistí jeden až tri riadky bez spinu
clearquad	Keď hráč vyčistí štyri alebo viac riadkov
finish	Keď hra skončí úspešne
death	Keď hra skončí neúspešne
i	Keď je možné počuť ďalšie dieliky a nasledujúci dielik je I
j	Keď je možné počuť ďalšie dieliky a nasledujúci dielik je J
l	Keď je možné počuť ďalšie dieliky a nasledujúci dielik je L
o	Keď je možné počuť ďalšie dieliky a nasledujúci dielik je O
s	Keď je možné počuť ďalšie dieliky a nasledujúci dielik je S
t	Keď je možné počuť ďalšie dieliky a nasledujúci dielik je T
z	Keď je možné počuť ďalšie dieliky a nasledujúci dielik je Z
countdown{číslo}	Keď tikne odpočet a zostáva číslo + 1 tikov
combo_{číslo}	Keď hráč vyčistí jeden až tri riadky bez spinu a combo reťazec má dĺžku rovnú číslu
combo_{číslo}_power	Keď hráč vyčistí štyri riadky alebo urobí spin a combo reťazec má dĺžku rovnú číslu

Pre kompletnú kontrolu nad prehrávaním zvukových efektov je možné napísať skript do súboru `soundEffects.lua`. Tento skript dokáže registrovať obslužné funkcie na herné

udalosti a v rámci obslužných funkcií hrať zvukové efekty, či už základné, alebo nahrané ako súbory v zložke zvukových efektov.

Registrovanie obslužných funkcií je vysvetlené v sekcii A.2.

Prehrávanie zvukových efektov je možné buď pomocou vrátenia názvu zvukového efektu na konci funkcie – keďže Lua podporuje vracanie viacerých hodnôt, je možné takto prehrať aj viac zvukových efektov naraz.

Ďalšou možnosťou prehrávania zvukových efektov v skripte je zavolanie funkcie `Play()`. Táto funkcia akceptuje jeden parameter – názov zvukového efektu, ktorý má byť prehraný. Zvukové efekty prehrané funkciou `Play()` alebo pomocou vrátenia z funkcie nijako neprerušujú predošlé zvukové efekty a môže ich hrať až 50 zároveň.

Poslednou možnosťou prehrávania zvukových efektov je zavolanie funkcie `PlayAsAnnouncer()`. Táto funkcia funguje podobne ako funkcia `Play()`, ale pomocou nej môže byť prehrávaný vždy len jeden zvukový efekt. Ak bude táto funkcia zavolaná počas toho, čo sa pomocou nej hrá nejaký zvukový efekt, tak sa práve hraný efekt zruší a miesto neho sa začne prehrávať nový špecifikovaný efekt.

A.6 Používanie vlastných skriptov pre správu hry

Skripty pre správu hry alebo „Game Manager skripty“ ovládajú skórovanie, úrovne, gravitáciu a iné časti hry. Pre vytvorenie a použitie vlastného Game Manager skriptu je treba nasledovať tieto kroky:

1. Vymyslieť názov Game Manageru a vytvoriť súbor s daným názvom a koncovkou `.lua`. Tento súbor treba vložiť do perzistentného herného úložiska, do zložky `rulesCustomization/gameManagers`.
2. Napísať v súbore skript, ktorý bude slúžiť ako daný Game Manager.
3. Vo vlastných herných nastaveniach, v sekcii „Objective“ vybrať hodnotu „Custom“ ako typ použitého Game Managera.
4. V ponuke vlastných Game Managerov vybrať vytvorený skript podľa názvu.

Obsah skriptu pre správu hry

Game Manager skript si dokáže zaregistrovať obslužné funkcie na akékoľvek udalosti, aké sa v hre dejú. Popis registrovania týchto funkcií je uvedený v sekcii A.2.

Ešte pred spustením užívateľského Game Manager skriptu sa v ňom sprístupnia nasledovné hodnoty:

- `StartingLevel` – hodnota obsahuje hráčom vybranú úroveň vo formáte `string`. V prípade, že daný skript nepodporuje úrovne alebo nepodporuje túto vybranú úroveň je možné ju jednoducho ignorovať.
- `Board` – herná plocha momentálne hranej hry. Táto premenná obsahuje nasledujúce hodnoty:
  - `Board.Width` – šírka hernej plochy v blokoch.
  - `Board.Height` – výška hernej plochy v blokoch.

- `Board.LethalHeight` – výška, ktorá bola v herných nastaveniach nastavená ako smrteľná. Je relevantná len ak je podporovaná mechanika lock out.
- `Board.GarbageHeight` – výška odpadových riadkov, ktoré sú momentálne na hernej ploche.
- `Board.Slots` – zoznam zoznamov boolovských hodnôt. Každý vnútorný zoznam je jeden riadok hernej plochy. V daných riadkoch sú dané hodnoty `true` v prípade, že dané miesto je obsadené blokom.

  Vonkajší zoznam je dynamický – vždy obsahuje len toľko hodnôt, aká je najväčšia výška momentálne položeného bloku na ploche.

  Keďže tento zoznam je objekt z C#, tak je na rozdiel od Lua tabuliek indexovaný od 0, nie od 1.

Takisto obsahuje nasledujúce funkcie:

- `Board:AddGarbageLayer(slotsTable, addToLast)` – funkcia, ktorá slúži na pridanie odpadových riadkov na hernú plochu.

  Argument `slotsTable` je tabuľka tabuliek boolovských hodnôt, ktoré sú `true` pre obsadený slot a `false` pre voľný slot.

  Argument `addToLast` je boolovská hodnota ktorá určuje, či bude daná vrstva odpadových riadkov spojená s predchádzajúcou. Toto je dôležité pre výzory blokov v spojenom formáte.
- `Board:ClearAllBlocks()` – funkcia, ktorá vyčistí všetky bloky na hernej ploche.

Okrem prístupných hodnôt sú v Game Manager skripte zároveň prístupné nasledovné volateľné funkcie:

- `EndGame()` – okamžite ukončí hru úspešne.
- `LoseGame()` – okamžite ukončí hru neúspešne. Pokiaľ je v herných nastaveniach nastavené, že prehrávanie je v poriadku, tak je toto rovnaké ako `EndGame`.
- `AddScore(score)` – pridá hodnotu k momentálnemu skóre. Skóre je na začiatku každej hry automaticky resetované na nula.

  Argument `score` je číselná hodnota, ktorá má byť pridaná k pôvodnej hodnote skóre.
- `SetLevel(level)` – zmení momentálnu úroveň.

  Argument `level` je hodnota, ktorá bude nastavená ako nová úroveň. Pri nastavovaní bude konvertovaná na typ `string`.
- `SetGravity(newGravity)` – zmení momentálnu gravitáciu v blokoch za 1/60 sekundy.

  Argument `newGravity` je nová hodnota gravitácie. Pri nastavovaní bude konvertovaná na číslo.
- `SetLockDelay(newDelay)` – zmení momentálnu lock delay v sekundách.

  Argument `newDelay` je nová hodnota lock delay. Pri nastavovaní bude konvertovaná na číslo.

- `SetGameEndCondition(current, total, name)` – Zmení momentálnu hodnotu podmienky pre ukončenie hry. Táto funkcia môže byť užitočná pre vlastné podmienky ukončenia hry.

  Argument `current` je momentálna nová hodnota podmienky pre ukončenie hry.

  Argument `total` je celková podmienka pre ukončenie hry.

  Argument `name` je názov tejto podmienky pre ukončenie hry.

- `SetLevelUpCondition(current, total, name)` – Zmení momentálnu hodnotu podmienky pre zmenu úrovne.

  Argument `current` je momentálna nová hodnota podmienky pre zmenu úrovne.

  Argument `total` je celková podmienka pre zmenu úrovne.

  Argument `name` je názov tejto podmienky pre zmenu úrovne.

A.7 Používanie vlastných generátorov odpadových riadkov

Skripty pre generovanie odpadových riadkov, alebo „Garbage Generator“ skripty ovládajú spôsob, akým sa budú na hernú plochu pridávať odpadové riadky. Pre vytvorenie a použitie vlastného Garbage Generator skriptu je treba nasledovať tieto kroky:

1. Vymyslieť názov Garbage Generatoru a vytvoriť súbor s daným názvom a koncovkou `.lua`. Tento súbor je treba vložiť do perzistentného herného úložiska, do zložky `rulesCustomization/garbageGenerators`.
2. Napísať v súbore skript, ktorý bude slúžiť ako daný Garbage Generator.
3. Vo vlastných herných nastaveniach, v sekcii „Objective“ vybrať hodnotu „Custom“ ako typ použitého Garbage Generatoru.
4. V ponuke vlastných Garbage Generatorov vybrať vytvorený skript podľa názvu.

Obsah skriptu pre generovanie odpadových riadkov

Garbage Generator skript na rozdiel od iných skriptov nedokáže registrovať obslužné funkcie na žiadne udalosti. Miesto toho jednoducho vracia dve funkcie, ktoré budú reagovať na požiadavky o generovanie odpadových riadkov a na požiadavky o resetovanie stavu generátora.

Ešte pred spustením užívateľského Garbage Generator skriptu sa v ňom sprístupní premenná `Board` – herná plocha momentálne hranej hry. Táto premenná obsahuje nasledujúce hodnoty:

- `Board.Width` – šírka hernej plochy v blokoch.
- `Board.Height` – výška hernej plochy v blokoch.
- `Board.LethalHeight` – výška, ktorá bola v herných nastaveniach nastavená ako smrteľná. Je relevantná len ak je podporovaná mechanika lock out.
- `Board.GarbageHeight` – výška odpadových riadkov, ktoré sú momentálne na hernej ploche.

- `Board.Slots` – zoznam zoznamov boolovských hodnôt. Každý vnútorný zoznam je jeden riadok hernej plochy. V daných riadkoch sú dané hodnoty `true` v prípade, že dané miesto je obsadené blokom.

  Vonkajší zoznam je dynamický – vždy obsahuje len toľko hodnôt, aká je najväčšia výška momentálne položeného bloku na ploche.

  Keďže tento zoznam je objekt z C#, tak je na rozdiel od Lua tabuliek indexovaný od 0, nie od 1.

Premenná Board takisto obsahuje nasledujúce funkcie:

- `Board:AddGarbageLayer(slotsTable, addToLast)` – funkcia, ktorá slúži na pridanie odpadových riadkov na hernú plochu.

  Argument `slotsTable` je tabuľka tabuliek boolovských hodnôt, ktoré sú `true` pre obsadený slot a `false` pre voľný slot.

  Argument `addToLast` je boolovská hodnota ktorá určuje, či bude daná vrstva odpadových riadkov spojená s predchádzajúcou. Toto je dôležité pre výzory blokov v spojenom formáte.

Pre generovanie odpadových riadkov a resetovanie stavu generátora je treba na konci skriptu vrátiť dve hodnoty – prvá hodnota musí byť funkcia, používaná na generovanie odpadových riadkov a druhá hodnota musí byť funkcia, používaná na resetovanie stavu.

Funkcia pre generovanie odpadových riadkov musí akceptovať dva parametre. Prvým parametrom je počet vyžiadaných odpadových riadkov a druhým parametrom je správa o práve položenom dieliku. Obsah danej správy je možné nájsť v sekcii A.2. Samotné generovanie odpadových riadkov je treba robiť pomocou funkcie `Board:AddGarbageLayer(slotsTable, addToLast)`, popísanej vyššie.

Funkcia pre resetovanie stavu generátoru neprijíma žiadne argumenty. Seed náhodného generátora bude nastavený automaticky, takže v tejto funkcii je treba len resetovať potrebné globálne premenné.

```
function GenerateGarbage(amount, message)
    -- this function will handle generating garbage
end

function Reset()
    -- this function will handle resetting the generator
end

return GenerateGarbage, Reset
```

Výpis A.4: Výpis znázorňuje príklad štruktúry skriptu pre generovanie odpadových riadkov.

A.8 Používanie vlastných generátorov dielikov

Skripty pre náhodné generovanie, alebo „Randomizer“ skripty ovládajú spôsob, akým sa budú vyberať nasledujúce dieliky. Pre vytvorenie a použitie vlastného Randomizer skriptu je treba nasledovať tieto kroky:

1. Vymyslieť názov Randomizeru a vytvoriť súbor s daným názvom a koncovkou `.lua`. Tento súbor je treba vložiť do perzistentného herného úložiska, do zložky `rulesCustomization/randomizers`.
2. Napísať v súbore skript, ktorý bude slúžiť ako daný Randomizer.
3. Vo vlastných herných nastaveniach, v sekcii „General“, vybrať hodnotu „Custom“ ako typ použitého Randomizera.
4. V ponuke vlastných Randomizerov vybrať vytvorený skript podľa názvu.

Obsah skriptu pre generovanie dielikov

Randomizer skript na rozdiel od iných skriptov nedokáže registrovať obslužné funkcie na žiadne udalosti. Miesto toho jednoducho vracia dve funkcie, ktoré budú reagovať na požiadavky o ďalší dielik a na požiadavky o resetovanie stavu generátora.

Ešte pred spustením užívateľského Randomizer skriptu sa v ňom sprístupní premenná `AvailablePieces`. Táto premenná obsahuje všetky typy dielikov, ktoré je možné generovať. Jej obsah by mal byť identický s definíciou premennej vo výpise A.5.

```
AvailablePieces = {"i", "o", "t", "l", "j", "s", "z"};
```

Výpis A.5: Výpis znázorňuje obsah premennej s generovateľnými dielikmi.

Pre generovanie dielikov a resetovanie stavu náhodného generátora je treba na konci skriptu vrátiť dve hodnoty – prvá hodnota musí byť funkcia, používaná na generovanie dielikov a druhá hodnota musí byť funkcia, používaná na resetovanie stavu. Základná štruktúra skriptu pre náhodný generátor je zobrazená na výpise A.6.

Funkcia pre generovanie dielikov neprijíma žiadne argumenty. Musí vrátiť jednu hodnotu typu `string`, ktorá identifikuje typ vygenerovaného dielika. Musí to byť jedna z hodnôt obsiahnutých v tabuľke `AvailablePieces` vysvetlenej vyššie.

Funkcia pre resetovanie stavu neprijíma žiadne argumenty. Seed náhodného generátora bude nastavený automaticky, takže v tejto funkcii je treba len resetovať potrebné globálne premenné.

```
function GetNextPiece()
    -- this function will handle generating the next piece
end

function Reset()
    -- this function will handle resetting the generator
end

return GetNextPiece, Reset
```

Výpis A.6: Výpis znázorňuje príklad štruktúry skriptu pre generovanie dielikov.

A.9 Používanie vlastných rotačných systémov

Rotačné systémy ovládajú kicky, ktoré budú použité pri rotácii jednotlivých dielikov. Takisto určujú počiatočný rotačný stav dielikov a kicky, ktoré budú brané ako výnimky pre

identifikovanie mini spinov ako plných spinov. Pre vytvorenie a použitie vlastného rotačného systému je treba nasledovať tieto kroky:

1. Vymyslieť názov rotačného systému a vytvoriť súbor s daným názvom a koncovkou `.json`. Tento s[bor je treba vložiť do perzistentného herného úložiska, do zložky `rulesCustomization/rotationSystems`.
2. Napísať v súbore konfiguráciu daného rotačného systému.
3. Vo vlastných herných nastaveniach, v sekcii „Controls“, vybrať hodnotu „Custom“ ako typ použitého rotačného systému.
4. V ponuke vlastných rotačných systémov vybrať vytvorený rotačný systém podľa názvu.

Ako napísať konfiguráciu rotačného systému

Rotačný systém je jednoducho zoznam tabuliek kickov a počiatočných rotačných stavov pre rôzne typy dielikov.

Počiatočný stav je typ enum a má 4 hodnoty:

- 0 - východzí rotačný stav.
- 90 - dielik je otočený o 90 stupňov proti smeru hodinových ručičiek.
- 180 - dielik je otočený o 180 stupňov.
- 270 - dielik je otočený o 90 stupňov po smere hodinových ručičiek.

Nastavenie počiatočného stavu zmení ako sa dielik objaví, ale nie ako bude otočený v ukážkach nasledujúcich dielikov.

Všetky kicky, ktoré sú v rotačnom systéme obsiahnuté, potrebujú byť v tvare objektov s hodnotami x a y. Hodnota x určuje o koľko sa dielik pohne na x osi a hodnota y určuje, o koľko sa dielik pohne na y osi. Osi začínajú od ľavého dolného rohu a rastú smerom vpravo hore.

Rotačný systém obsahuje tabuľky pre konkrétne dieliky a jednu východziu tabuľku, ktorá sa bude používať v prípade, že sa nenájde konkrétna tabuľka pre typ dielika. Väčšinou je používaná jedna tabuľka pre dielik I a jedna východzia tabuľka pre všetky ostatné dieliky.

Príklad jednoduchej konfigurácie vlastného rotačného systému je uvedený na výpise A.7.

```
{
    "KickTables": {
        "i": {
            "StartState": 180,
            "FullSpinKicks": [],
            "ZeroToThree": [{"x": 0, "y": 0}]
        }
    }
    "DefaultTable": {
        "StartState": 0,
        "FullSpinKicks": [{"x": -1, "y": -2},{"x": 1, "y": -2}],
        "ZeroToThree": [{"x": 0, "y": 0}],
```

```
        "ZeroToOne": [{"x": 0, "y": 0}],
        "OneToZero": [{"x": 0, "y": 0}],
        "OneToTwo": [{"x": 0, "y": 0}],
        "TwoToOne": [{"x": 0, "y": 0}],
        "TwoToThree": [{"x": 0, "y": 0}],
        "ThreeToTwo": [{"x": 0, "y": 0}],
        "ThreeToZero": [{"x": 0, "y": 0}],
        "ZeroToTwo": [{"x": 0, "y": 0}],
        "TwoToZero": [{"x": 0, "y": 0}],
        "OneToThree": [{"x": 0, "y": 0}],
        "ThreeToOne": [{"x": 0, "y": 0}],
    }
}
```

Výpis A.7: Výpis zobrazuje zjednodušený príklad konfigurácie rotačného systému.

A.10 Používanie počítadiel štatistík

V časti globálnych herných nastavení s názvom „Stat Counting“ je možné vidieť vlastné skupiny počítadiel štatistík. Každý herný mód má nastavenú skupinu, ktorá sa preň berie ako východzia a v herných nastaveniach sa dá zmeniť na jednu z vlastných skupín, ktoré sa nachádzajú v globálnych nastaveniach.

V týchto nastaveniach je možné vytvoriť ľubovoľné množstvo skupín, pomenovať ich tak, aby popisovali, k čomu sú určené a definovať v nich niekoľko rôznych počítadiel, ktoré môžu byť buďto vstavané do samotnej hry, alebo naprogramované a vložené do perzistentného herného úložiska, do zložky `statCounters`.

Vo vlastných skupinách počítadiel štatistík sa pre každé počítadlo dá nastaviť niekoľko hodnôt:

- Pozícia – pozícia v blokoch, relatívna k ľavému dolnému rohu hernej plochy.
- Veľkosť – veľkosť v blokoch. Veľkosť rastie smerom vpravo hore.
- Obnovovací interval – pokiaľ sa počítadlo štatistík potrebuje obnovovať periodicky, tak toto je interval, v akom bude dostávať periodické požiadavky na obnovenie.
- Typ počítadla – určuje, či bude počítadlo jedno zo vstavaných, alebo vlastné počítadlo. Vstavané počítadlá sú:
  - Stopwatch – počíta ubehnutý čas. Toto počítadlo je treba obnovovať periodicky.
  - Lines Cleared – počíta vyčistené riadky.
  - Pieces – počíta položené dieliky a rýchlosť, akým sa dieliky pokladajú. Kvôli rýchlosti je toto počítadlo treba obnovovať periodicky.
  - Score – počíta skóre a priemerné skóre na počet položených dielikov.
  - Inputs – počíta stlačené tlačidlá.
  - Level – ukazuje úroveň a priebeh podmienky pre zmenu úrovne.
  - All Clears – počíta vykonané all cleary.
  - Holds – počíta vykonané holdy.

- Remaining End Condition – počíta podmienku pre ukončenie hry.
- All Clears Action Text – ukazuje akčný text, keď nastane all clear.
- Back-to-back Action Text – ukazuje akčný text, keď nastane back-to-back vyčistenie.
- Clear Type Action Text – ukazuje akčný text, keď nastane akékoľvek vyčistenie.
- Combo Action Text – ukazuje akčný text podľa stavu comba.

Okrem týchto počítadiel je možné vybrať aj vlastné počítadlo. V tomto prípade treba špecifikovať aj názov súboru, v ktorom sa počítadlo nachádza. Brané sú len počítadlá, ktoré sa nachádzajú v perzistentnom hernom úložisku v zložke `statCounters`. Ak sa vlastné počítadlo nájde a načíta, hra ukáže upozornenie o úspechu a ak sa počítadlo nenájde alebo nebude možné ho načítať, hra ukáže upozornenie o neúspechu.

Okrem globálnych nastavení je takisto možné meniť niektoré parametry počítadiel priamo počas hry. Pokiaľ nad daným počítadlom hráč podrží kurzor, v rohoch počítadla sa objavia držadlá, pomocou ktorých je možné meniť pozíciu a veľkosť daného počítadla.

Používanie vlastných počítadiel

Vlastné počítadlá sú skripty, ktoré si dokážu zaregistrovať herné udalosti a reagovať na ne zmenou textu, farbami a viditeľnosťou. Pre vytvorenie a použitie vlastného počítadla štatistík je treba nasledovať tieto kroky:

1. Vymyslieť názov počítadla a vytvoriť súbor s daným názvom a koncovkou `.lua`. Tento súbor je treba vložiť do perzistentného herného úložiska, do zložky `statCounters`.
2. Napísať v súbore skript, ktorý bude slúžiť ako dané počítadlo.
3. V globálnych nastaveniach, v sekcii „Stat Counting“, buď vytvoriť novú skupinu počítadiel alebo vybrať nejakú skupinu a pridať do nej nové počítadlo.
4. Pre pridané počítadlo vybrať hodnotu „Custom“ ako typ počítadla a ako napísať názov súboru počítadla do poľa súboru. Názov treba písať aj s koncovkou.

Obsah skriptu vlastného počítadla

Skript počítadla si dokáže zaregistrovať obslužné funkcie na akékoľvek udalosti, aké sa v hre dejú. Popis registrovania týchto funkcií je uvedený v sekcii A.2.

Okrem registrovania štandardných herných udalostí je možné takisto zaregistrovať udalosť `CounterUpdated`. Táto funkcia bude volaná každý časový interval, špecifikovaný v nastaveniach počítadla. Pokiaľ bude tento časový interval mať hodnotu 0, tak sa bude táto funkcia volať každý snímok hry. Udalosť `CounterUpdated` na rozdiel od ostatných herných udalostí neposkytuje žiadnu správu s informáciami.

Ešte pred spustením užívateľského skriptu počítadla sa v ňom sprístupnia nasledovné hodnoty:

- `StatUtility` – C# objekt, ktorý obsahuje rôzne funkcie, ktoré pomáhajú s formátovaním textu a inými vecami. Obsiahnuté funkcie sú nasledovné:

  - `StatUtility:GetCurrentTime()` – jednoducho vráti momentálny herný čas ako číslo. Tento čas sa nemusí zhodovať s časom správ, ktoré počítadlo štatistík dostane pre obsluhovanie udalostí.
  - `StatUtility:GetFormattedTime()` – vráti momentálny herný čas ako pekne sformátovaný reťazec znakov.
  - `StatUtility:FormatTime(seconds, showMilliseconds)` – skonvertuje argument `seconds` na číslo a vráti ho sformátované ako čas, rovnako ako funkcia `GetFormattedTime()`.

    Hodnota argumentu `seconds` sa berie ako množstvo sekúnd.

    Argument `showMilliseconds` určuje, či sa vo formátovanom reťazci majú zobrazovať milisekundy. Východzia hodnota tohoto argumentu je `true`.
  - `StatUtility:FormatNumber(number, decimals)` – skonvertuje argumenty na čísla a vráti hodnotu argumentu `number` zaokrúhlenú na `decimals` desatinných miest ako sformátovaný reťazec znakov.

    V prípade, že číslo je nekonečno, sa vráti hodnota `"+INF"` alebo `"-INF"`. V prípade, že hodnota argumentu `number` nie je číselná, vráti sa hodnota `"NaN"`.

    Argument `decimals` bude obmedzený do číselného intervalu nula až pätnásť.

- `Stats` – C# objekt, ktorý obsahuje hodnoty natívne počítaných štatistík. aby sa obmedzilo zbytočné počítanie štatistík je väčšinou možné jednoducho zobrazovať hodnoty z tohoto objektu. Štatistiky, ktoré sú v tomto objekte prístupné:
  - `Stats.Score` – skóre momentálnej hry. Typ `long`.
  - `Stats.Level` – momentálny level. Typ `string`.
  - `Stats.LinesCleared` – počet vyčistených riadkov. Typ `uint`.
  - `Stats.PiecesPlaced` – počet položených dielikov. Typ `uint`.
  - `Stats.KeysPressed` – počet stlačených tlačidiel. Typ `uint`.
  - `Stats.Singles` – počet jedno-riadkových vyčistení. Typ `uint`.
  - `Stats.Doubles` – počet dvoj-riadkových vyčistení. Typ `uint`.
  - `Stats.Triples` – počet troj-riadkových vyčistení. Typ `uint`.
  - `Stats.Quads` – počet štvor-riadkových vyčistení. Typ `uint`.
  - `Stats.Spins` – počet spinov bez vyčistení. Typ `uint`.
  - `Stats.MiniSpins` – počet mini spinov bez vyčistení. Typ `uint`.
  - `Stats.SpinSingles` – počet spinov s jedno-riadkovými vyčisteniami. Typ `uint`.
  - `Stats.SpinDoubles` – počet spinov s dvoj-riadkovými vyčisteniami. Typ `uint`.
  - `Stats.SpinTriples` – počet spinov s troj-riadkovými vyčisteniami. Typ `uint`.
  - `Stats.SpinQuads` – počet spinov s štvor-riadkovými vyčisteniami. Typ `uint`.
  - `Stats.MiniSpinSingles` – počet mini spinov s jedno-riadkovými vyčisteniami. Typ `uint`.
  - `Stats.MiniSpinDoubles` – počet mini spinov s dvoj-riadkovými vyčisteniami. Typ `uint`.
  - `Stats.MiniSpinTriples` – počet mini spinov s troj-riadkovými vyčisteniami. Typ `uint`.

  - `Stats.MiniSpinQuads` – počet mini spinov s štvor-riadkovými vyčisteniami. Typ `uint`.
  - `Stats.LongestCombo` – dĺžka najdlhšieho combo reťazca počas tejto hry. Typ `uint`.
  - `Stats.LongestBackToBack` – dĺžka najdlhšieho back-to-back reťazca počas tejto hry. Typ `uint`.
  - `Stats.AllClears` – počet vykonaných all clearov. Typ `uint`.
  - `Stats.Holds` – počet vykonaných holdov. Typ `uint`.
  - `Stats.GarbageLinesCleared` – počet vyčistených odpadových riadkov. Typ `uint`.
  - `Stats.PiecesPerSecond` – priemerné množstvo dielikov položených za sekundu. Typ `double`.
  - `Stats.KeysPerPiece` – priemerné množstvo stlačených tlačidiel na položenie jedného dielika. Typ `double`.
  - `Stats.KeysPerSecond` – priemerné množstvo stlačených tlačidiel za sekundu. Typ `double`.
  - `Stats.LinesPerMinute` – priemerné množstvo vyčistených riadkov za minútu. Typ `double`.
- `Board` – herná plocha momentálne hranej hry. Táto premenná obsahuje nasledujúce hodnoty:
  - `Board.Width` – šírka hernej plochy v blokoch.
  - `Board.Height` – výška hernej plochy v blokoch.
  - `Board.LethalHeight` – výška, ktorá bola v herných nastaveniach nastavená ako smrteľná. Je relevantná len ak je podporovaná mechanika lock out.
  - `Board.GarbageHeight` – výška odpadových riadkov, ktoré sú momentálne na hernej ploche.
  - `Board.Slots` – zoznam zoznamov boolovských hodnôt. Každý vnútorný zoznam je jeden riadok hernej plochy. V daných riadkoch sú dané hodnoty `true` v prípade, že dané miesto je obsadené blokom.

    Vonkajší zoznam je dynamický – vždy obsahuje len toľko hodnôt, aká je najväčšia výška momentálne položeného bloku na ploche.

    Keďže tento zoznam je objekt z C#, tak je na rozdiel od Lua tabuliek indexovaný od 0, nie od 1.

Okrem týchto hodnôt sú v skripte počítadla prístupné aj nasledujúce funkcie:

- `SetText(text)` – nastaví text počítadla na hodnotu argumentu `text`.
- `SetColor(color)` – nastaví farbu textu počítadla na hodnotu argumentu `color`. Akceptované formáty farby sú štandardné HTML farby a hexadecimálne hodnoty v tvare `#rrggbb` alebo `#rrggbbaa`. V prípade zadania neplatnej hodnoty bude použitá biela farba.

  Pokiaľ bola farba pred zavolaním tejto funkcie animovaná, tak sa animácia zruší a farba zostane na tejto hodnote.

- `SetVisibility(visibility)` – nastaví viditeľnosť textu počítadla na hodnotu argumentu `visibility`. Hodnota tohoto argumentu byť číslo medzi 0 a 1.

  Pokiaľ bola viditeľnosť pred zavolaním tejto funkcie animovaná, tak sa animácia zruší a viditeľnosť zostane na tejto hodnote.

- `AnimateColor(color, duration)` – animuje farbu textu počítadla z farby, ktorú počítadlo malo počas zavolania tejto funkcie na hodnotu argumentu `color`. Táto animácia bude trvať `duration` sekúnd.

- `AnimateVisibility(visibility, duration)` – animuje viditeľnosť textu počítadla z viditeľnosti, ktorú počítadlo malo počas zavolania tejto funkcie na hodnotu argumentu `visibility`. Táto animácia bude trvať `duration` sekúnd.

- `SetAlignment(alignment)` – nastaví zarovnanie textu. Podporované hodnoty sú: `"left"`, `"right"`, `"center"`, `"top"`, `"bottom"`, `"middle"`.

- `SetTextSize(textSize)` – nastaví veľkosť textu počítadla v blokoch. Bez zavolania tejto funkcie je veľkosť počítaná automaticky tak, aby text vyplnil celú plochu počítadla.

Okrem používania funkcie `SetText(text)` na nastavenie textu počítadla je možné aj z obslužných funkcií udalostí jednoducho vrátiť `string` hodnotu. Pokiaľ takáto hodnota vrátená nebude, tak text počítadla zostane rovnaký.