FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Vesmírné vylomeniny: softwarové řešení hry na interaktivní stěnu
Student:	Matěj Chlan
Vedoucí:	Ing. Radek Richtr, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Softwarové inženýrství 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Práce má za úkol prozkoumat stávající stav aplikacích vhodných primárně (ale nejen) na interaktivní dotykovou stěnu umístěnou v laboratoři ggLab a provést finalizaci projektu "Spaceship Shenanigans" - hry pro několika hráčů vytvořené na tuto interaktivní dotykovou stěnu.

1) Proveďte rešerši návrhových vzorů vhodných pro vývoj her
2) Analyzujte již vytvořený prototyp hry jak z hlediska kódu, tak z hlediska herních mechanik
3) Na základě analýzy vytvořte game design dokument
4) Na základě analýzy proveďte softwarový návrh hry
5) Na základě předchozích bodů vytvořte implementaci hry
6) Hru řádně otestujte, včetně uživatelských testů

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 5. února 2024 v Praze.

Bakalářská práce

VESMÍRNÉ VYLOMENINY: SOFTWAROVÉ ŘEŠENÍ HRY NA INTERAKTIVNÍ STĚNU

Matěj Chlan

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Radek Richtr Ph.D.
16. května 2024

Odkaz na tuto práci: Chlan Matěj. *Vesmírné vylomeniny: softwarové řešení hry na interaktivní stěnu.* Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Obsah

Seznam obrázků

Seznam výpisů kódu

Především bych chtěl poděkovat panu Ing. Radku Richtrovi, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce. Dále děkuji mé přítelkyni Magdaléně Musilové, za podporu nejen při tvorbě této práce, ale i při celém studiu. Nakonec bych rád poděkoval Anně Musilové za spolupráci a část vizuální stránky hry.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. Dále prohlašuji, že jsem s Českým vysokým učením technickým v Praze uzavřel dohodu, na jejímž základě se ČVUT vzdalo práva na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona. Tato skutečnost nemá vliv na ust. § 47b zákona č. 111/1998 Sb., o vysokých školách, ve znění pozdějších předpisů

V Praze dne 16. května 2024

Abstrakt

Tato práce se zabývá tvorbou hry pro interaktivní stěnu vyvíjené v Unity. Hlavním cílem je vylepšit stávající hru, jejíž kód a ovládání nejsou v optimálním stavu. Právě tyto prvky jsou hlavními body analýzy. Ta dále zkoumá i jiné hry dostupné pro interaktivní stěnu. Pro zajištění kvalitního softwarového návrhu je provedena rešerše návrhových vzorů. Následně je vytvořen softwarový a herní návrh, který slouží jako základ pro implementaci.

Klíčová slova interaktivní stěna, hra, Unity, návrhové vzory, ovládání

Abstract

This thesis deals with the creation of an interactive wall game developed in Unity. The main goal is to improve an existing game whose code and controls are not in optimal condition. It is these elements that are the main points of the analysis. Thesis also explores other games available for the interactive wall. Design patterns are researched to ensure high-quality software design. Subsequently, a software and game design document is created, which serves as the basis for the implementation.

Keywords interactive wall, game, Unity, design patterns, controls

Seznam zkratek

FPS	frames per second
GDD	game design dokument
LED	Light-Emitting Diode
OOP	objektově orientované programování
RGB	red, green, blue

Úvod

Z mé zkušenosti z fakultních dnů otevřených dveří mnoho zájemců o studium chce nastoupit kvůli jejich vztahu k videohrám. Někteří mají vášnivý zájem o vývoj her, ať už sami něco umí, či chtějí začít od nuly. Jiní zas pouze rádi tráví čas hraním a tak vidí informační technologie jako rozumnou volbu pro pokračování studia. Tito studenti mohou navštívit laboratoř ggLab ve 14. patře budovy A, kde na dnech otevřených dveří bývají prezentovány studentské hry. Vytvořené jako semestrální či bakalářské práce nebo na školou organizovaném GameJamu, tyto hry jsou téměř všechny vytvořené pro počítače. Laboratoř ovšem disponuje rozměrnou interaktivní stěnou. S podobnou technologií se lidé moc často nesetkají. Je tedy škoda, že její potenciál dlouho nebyl využit. Samozřejmě s hardwarem byl dodán i software. Initi Playground slouží jako spouštěč her či aplikací a s několika hrami byl dodán. Všechny ale mají spíše jednoduché mechaniky, kde se hráči dotknou nějakého objektu a ten se zničí.

To jsme chtěli v rámci předmětu Softwarový týmový projekt 1 (BI-SP1) změnit. V týmu šesti lidí jsme jako semestrální práci vytvořili hru Spaceship Shenanigans. Naším cílem bylo udělat zábavný, netriviální zážitek pro interaktivní stěnu. Z časových a organizačních důvodů jsme ale nestihli realizovat všechny nápady. Kvůli nedostatku zkušeností s vývojem her a s touto novou technologií je kód špatně navrhnut a obtížně rozšiřitelný. Vzhledem k mému zájmu o vývoj videoher jsem se rozhodl hru přepracovat a sestavit ji znovu od základu. Mým záměrem bylo vytvořit kvalitní hru, která bude poutat zájemce o studium.

Na hře se mnou pracovala Anna Musilová. Cílem její bakalářské práce bylo vytvořit veškerou grafiku hry a uživatelské rozhraní. V průběhu vývoje se ale rozhodla pro ukončení spolupráce a zvolila si jiné téma pro závěrečnou práci. I přesto hře poskytla část vizuální stránky.

Hlavním cílem bakalářské práce je tedy přepracovat a dokončit hru na interaktivní stěnu. Původní verze byla vytvořena jako semestrální práce v předmětu BI-SP1. I přes to, že je hra funkční, má mnoho nedostatků, které brání jejímu využití například pro zaujmutí návštěvníků na veřejných akcích na fakultě. Důležitou vlastností práce je čitelnost a rozšiřitelnost kódu. Z tohoto důvodu bude provedena rešerše návrhových vzorů vhodných pro vývoj her. Dále bude provedena analýza Spaceship Shenanigans a dalších her dostupných na interaktivní stěně v laboratoři ggLab. Pro dokumentaci herního designu bude vytvořen game design document. Nakonec bude provedeno uživatelské testování, sloužící k hodnocení stavu hry a porovnání s původní verzí.

Kapitola 1
Návrhové vzory

V této kapitole byly prozkoumány návrhové vzory vhodné pro vývoj her. Návrhový vzor popisuje problém opakovaně se vyskytující v objektově orientovaném softwaru. Poté ukazuje jeho řešení. Tím však není konkrétní implementace, ale jakási obecná šablona, kterou lze ke specifickému řešení využít. Tato obecnost umožňuje návrhovým vzorům být využívané v mnoha situacích [1]. Použití některých návrhových vzorů je zjevné, u jiných byly přidány ilustrační příklady jejich použití. Definice a příklady využití v této kapitole byly z velké části převzaty z knihy *Game Programming Patterns* od Roberta Nystroma [2].

1.1 Game Loop

Textové hry typicky pouze reagují na vstup uživatele. Po zadání příkazu se provede výpočet, stav hry se změní a před hráče je postaveno nové rozhodnutí. V tuto chvíli se program uspí a čeká na další příkaz. Takový přístup je pro textové hry vhodný, ale nestačí pro hry, které obsahují systémy běžící nezávisle na vstupu hráče jako animace, zvuky či real-time gameplay. Ani tahové hry nemusejí čekat na hráče a v mezičase například přehrávají animace.

Tento problém lze vyřešit návrhovým vzorem Game Loop, jehož cílem je „oddělení rychlosti času virtuálního světa od vstupu uživatele a rychlosti procesoru“ [2]. Jak už název napovídá, jádrem tohoto vzoru je smyčka. Z ní je volán téměř všechen kód, který aplikace vykonává. V průběhu jedné iterace se zpracuje vstup, pokud nějaký byl, herní čas se posune o kousek dopředu, změní se stav virtuálního světa a nakonec se vše vykreslí na obrazovku. Protože každá iterace právě jednou aktualizuje obraz, říkáme jí snímek (angl. frame). Systém v tuto chvíli už není navázán na vstup, ale zatím je stále závislý na rychlosti procesoru. Výkonnější počítače zvládnou výpočet snímku za kratší dobu a hráči se bude jevit tempo herního světa vyšší.

To lze opravit dvěma způsoby. První je zvolit si fixní délku snímku. Pro jednoduchost uvažujme, že cílíme na 60 FPS. Požadovaná délka snímku je tedy 16 ms. Po dokončení výpočtů program uspíme tak, aby následující snímek začal právě 16 ms po začátku stávajícího. Výhodou tohoto přístupu je, že víme přesně délku snímku a logika hry s tím může počítat. Každý snímek tedy posune herní čas o 16 ms dopředu a zároveň trvá 16 ms reálného času. Co se ale stane, když výpočet bude trvat déle? Celkem přirozeně se hra zpomalí a hráč uvidí pokles FPS.

Druhý způsob si s touto komplikací poradí. Namísto fixní délky necháme snímku délku proměnnou. Pokud výpočet trvá déle, posuneme herní čas o delší úsek dopředu. Naopak rychlejší snímky změní stav jen malinko. Zde ale narážíme na další problém. Předem se neví, jak dlouho bude výpočet snímku trvat. Proto místo toho, abychom objekty informovali o délce současného snímku, předáme jim čas snímku minulého. [2]

Důležitost a potřeba tohoto vzoru ve videohrách vyplývá už jen z faktu, že mnoho herních en-

ginů má tuto smyčku v nějaké formě zabudovanou. Unity využívá kombinaci obou výše zmíněných způsobů časování ve formě dvou souběžných smyček. Jedna slouží především pro výpočty fyzikálního systému a má fixní rozestup mezi iteracemi. Pro provedení vlastního kódu v této smyčce se využívá metoda `FixedUpdate`. Druhá je využívaná k ostatní logice a vykreslování obrazu. Tato smyčka se opakuje co nejčastěji může a volá všechny metody `Update`. [3]

Díky silnému provázání tohoto vzoru a Update Method lze usoudit, že Unreal engine také využívá tento návrhový vzor. Metoda `AActor::Tick` je volána enginem každý snímek a jako parametr přijímá `DeltaSeconds`. To je délka minulého snímku upravená o škálování času [4]. Herní engine Godot dokonce umožňuje programátorům vytvořit si vlastní herní smyčku. Stačí vytvořit skript, který dědí z `MainLoop` [5].

1.2 Update Method

Update Method je návrhový vzor silně spjatý s Game Loop (sekce 1.1). Jeho cílem je rozdělit kód chování herních objektů na snímky. Nystrom [2] ho ukazuje v kontrastu s naivním řešením, kde se snažíme naprogramovat kostlivce tak, aby hlídkoval mezi dvěma body. Kód posouvání nepřítele napíšeme přímo do hlavní smyčky hry. Problém je očividně v rozšiřitelnosti kódu. Každý objekt ve hře by měl kód svého chování umístěný v jedné funkci, ze které se ihned stane nečitelná změť podmínek a příkazů.

Místo tohoto prvoplánového řešení využijeme Update Method. Scéna, či herní svět, si bude udržovat seznam objektů, které budou implementovat metodu Update. Herní smyčka pak každý snímek tuto metodu zavolá u všech objektů [2]. Tím jsme rozdělili práci do jednotlivých objektů, což je z hlediska OOP přirozenější.

Výhodou tohoto přístupu je pozorovatelná souběžnost práce herních prvků i přes to, že výpočet objektů není skutečně souběžný. Veškerá práce se děje na jednom vlákně[1]. Díky tomu nenastávají problémy typu „Race condition“ a nemusí se řešit zamykání proměnných. Avšak programátor by měl mít na paměti, že díky postupnému volání objekt nemusí vidět stav na začátku snímku. Pokud budeme nejprve aktualizovat objekt A a až poté objekt B, B uvidí stav herního světa po tom, co ho A změnil.

Nystrom[2] tvrdí, že tento sekvenční přístup je stejně lepší. Uvádí, že hra využívající Update Method je v jistém smyslu stále tahová. Z tohoto pohledu je snímek kompletní tah skládající se z postupného pohybování (volání metody Update) jednotlivými figurkami (herními objekty). Tyto pohyby přesouvají herní svět z jednoho korektního stavu do druhého. Problém souběžného výpočtu lze přirovnat k šachům, kde černý i bílý táhnou současně. Pokud oba v jednom tahu vstoupí na stejné políčko, tak je stav hry nevalidní.

Jak je napsáno výše, vzor Update Method je velmi provázaný s Game Loop. Smyčka volá metody Update u všech objektů každý snímek. Herní enginy toto typicky zajišťují za programátory, kteří se poté mohou zajímat o implementaci samotného chování. Unity nazývá tuto metodu jednoduše `Update` [6], Unreal má `AActor::Tick` [4] a Godot `_process` [7].

1.3 Component

Herní objekt typicky dělá mnoho věcí. Například nepřítel potřebuje umělou inteligenci k rozhodování, fyziku pro pohyb, grafiku a mnoho dalšího. Pokud bude reprezentován jednou třídou, tak ta bude muset zaopatřit všechny jeho různorodé potřeby. V průběhu vývoje vede tento přístup k dlouhým a propleteným třídám, kde každá změna je velmi náročná a kód se často opakuje.

Opakování kódu lze mnohdy eliminovat pomocí dědění. Základní třída poskytuje implementaci nějaké funkcionality a odvozená třída ji využívá. Dědění je relace typu is-a, tedy objekt je něčím. Problém ale nastává, když chceme vytvořit objekt, který má zaopatřovat mnoho odlišných

[1] Hra samozřejmě může využívat více vláken i s tímto návrhovým vzorem. To ale není v tuto chvíli podstatné.

funkcionalit. Například goblin je nepřítel, entita, sprite, fyzikální objekt, herní objekt a mnoho dalšího. Aby goblin dědil ze všech těchto tříd, pozbývá smysl. Vytváří se tím komplikovaná hierarchie, která využívá vícenásobného dědění[2]. Také se tato třída stává závislá na implementaci rodičů. Z těchto a mnoha dalších důvodů se doporučuje využívat kompozici před dědičností.[1]

To přesně zajišťuje návrhový vzor Component. Namísto stavění komplexních hierarchií, či psaní obrovských tříd, rozděluje kód do menších ucelených částí nazývaných komponenty [2]. V případě našeho výše zmíněného goblina každá část, kterou je, změníme na část, kterou má. Relace se změní z is-a na has-a. Goblin stále může být třída odvozená, třeba z herního objektu. Ostatní části ale oddělíme do vlastních tříd. Ty reprezentují samostatnější díl funkcionality, jako vykreslování grafiky nebo fyzikální reprezentace objektu. Slovo „samostatnější“ jsem zvolil z důvodu, že komponenty nemusejí být úplně samostatné a mohou na sobě záviset či spolupracovat s jinými komponenty. Například umělá inteligence, která rozhoduje o chování, může spolupracovat s fyzikou, která objektem hýbe.

Využití návrhového vzoru Component je ve hrách docela podstatné. Díky rozdělení kódu se zvyšuje čitelnost, znovupoužitelnost a především samostatnost nesouvisejících částí programu. Herní enginy podporují kompozici a jednotlivé herní objekty jsou tvořeny především komponenty. V Unity jsou komponenty nazývané Components a umisťují se na herní objekty, jejichž veškeré chování je diktováno právě komponenty. [8]

Unreal engine využívá komponenty podobně jako Unity. Nemohou existovat samostatně a tvoří chování herních objektů (nazývaných „Actor“). Rozdílem je možnost naprogramování samotného actora. Ten využívá komponentů pro dosáhnutí svého cíle. [9]

Herní engine Godot nemá přímo komponenty, ale jejich principů nabývá lehce jiným způsobem. Namísto toho, aby herní objekt obsahoval několik skriptů, je tvořen právě jedním. Kompozice pak vzniká jejich spojováním do stromových struktur. [10]

1.4 Prototype

Cílem návrhového vzoru Prototype je vytváření nových objektů pomocí již existujících. Hry často vytváří mnoho kopií nepřátel, předmětů či terénu. V tuto chvíli se pro jednoduchost zaměřme na nepřátele. V našem případě bude hra mít základny, které budou vytvářet monstra. Tedy budeme mít základnu duchů, základnu démonů a základnu kostlivců. Protože každá tvoří jiné objekty a ty potřebují různá nastavení, uděláme pro každou z nich samostatnou třídu. Těch ve výsledku vznikne velmi mnoho i přes to, že jejich činnost je v podstatě stejná.

Návrhový vzor prototype k tomuto problému přistupuje jinak. Jak již název napovídá, místo vytváření nových objektů pomocí konstruktoru, se budou používat prototypy. Prototyp je existující objekt, který má metodu `Clone`. Ta vytvoří jeho exaktní kopii. Díky tomu, že je tato metoda umístěna přímo ve třídě objektu, má přístup ke všem jeho proměnným. I těm, které nejsou z vnějšku dostupné. Pokud se vrátíme k příkladu se základnami nepřátel, tak můžeme kód upravit s použitím tohoto návrhového vzoru. Místo mnoha vytvoříme pouze jednu třídu `Spawner`. Její instance bude mít uloženou referenci na prototyp a ten při každém požadavku na vytvoření monstra naklonuje pomocí metody `Clone`. [2]

Síla tohoto návrhového vzoru se skrývá v jeho přístupu k vnitřnímu stavu objektu. Protože nevytváříme jen objekt s výchozími hodnotami, ale s určitým stavem, můžeme tím jednoduše zajistit variaci. Na začátku hry vytvoříme několik prototypů s odlišnými vlastnostmi. Budeme mít rychlé, pomalé, silné a slabé monstrum. Ty pak budou využity jako předloha pro jejich základny.

V knize *Level up your code with game programming patterns* [11] autor uvádí, že Unity poskytuje vývojářům jednoduché využití návrhového vzoru Prototype. Prefab je herní objekt, který lze sestavit v editoru. Může obsahovat komponenty a mít dětské objekty. Prefab je poté možné umístit do scény, nebo dynamicky vytvářet jeho instance ve hře. Instancování se provádí metodou

[2] To přináší své potíže typu Diamantový problém.

`Instantiate`. Co už ale autor nezmiňuje je fakt, že pomocí této metody lze duplikovat jakýkoliv objekt ve scéně. To lze díky tomu, že prefab i herní objekt jsou typu `GameObject`. [12]

Herní engine Godot poskytuje podobnou funkcionalitu jako Unity. Vývojář může uložit scénu, ta se serializovaná uloží do souboru a pak může být mnohokrát instancována. Jednotlivé scény mohou být vkládány do sebe a tím tvořit komplikovanější struktury, než základní návrhový vzor Prototype umožňuje. [13]

1.5 Observer

Návrhový vzor Observer mění směr závislosti mezi třídami. Při klasickém imperativním přístupu se informace o události přenese od zdroje k přijímači jednoduchým voláním funkce. Pokud ale zdroj nutně nevyžaduje znalost přijímače, vzniká zbytečná závislost. Tento problém řeší návrhový vzor Observer tak, že přijímač se přihlásí k odběru změn stavu zdroje. Zdroj pak pouze oznamuje změny dynamickému seznamu přijímačů aniž by při kompilaci bylo potřeba znát, které to budou. [2]

Příkladem využití může být tutoriál. Hráče chceme naučit skákat a když vyskočí, tak se posunout k další části výuky. Prvoplánové řešení by bylo do kódu zajišťujícího skok hráče přidat `tutorial.PlayerJumped()`. Nyní je tento kód volán při každém skoku, i když je potřeba jen jednou. Hráč musí mít referenci na tutoriál, který ve scéně ani nemusí být. Na to je nutné přidat podmínku. Ale snad největší problém je, že na skok může být mnoho dalších reakcí. Přehrání zvuku a animace, vytvoření obláčku prachu, zvýšení čítače skoků pro závěrečné statistiky a jiné. Z kódu pro skákání se stal kód volající mnoho funkcí v mnoha různých systémech.

Návrhový vzor Observer má pro takový problém elegantní řešení. Místo natvrdo napsaného kódu, který říká co se má stát, poskytneme rozhraní, přes které se mohou posluchači přihlásit k odběru události. Když pak nastane změna stavu, zdroj pošle oznámení všem přijímačům.

Observer bývá zakomponován do vyšších jazyků. V jazyce Java lze využít java.util.Observer. C# zas poskytuje klíčové slovo event. Výpis kódu 1 je příklad použití Observeru v Unity, které pro tento návrhový vzor využívá třídu UnityEvent. Ta má metody `AddListener` a `RemoveListener` pro přidávání, respektive odebírání posluchačů [14].

1.6 Singleton

Knižní definice záměru návrhového vzoru Singleton zní *„Zajišťuje, že třída má pouze jednu instanci a poskytuje k ní globální přístup“*. [1] Existují dva základní způsoby, jak těmto požadavkům dostát. První je vytvořit statickou třídu. Ta slouží jako jakýsi obal pro skupinu globálních proměnných a metod. Z principu nemůže mít více instancí a lze k ní přistupovat odkudkoliv z kódu. Definici Singletonu tedy splňuje, ale nemusí to vždy být vhodné řešení. Statické třídy neumožňují polymorfismus.

Druhý způsob je více flexibilní. Třída má opravdovou (ne statickou) instanci, ke které lze globálně přistupovat. Reference na ní je uložená ve statické proměnné. Jednoduchá implementace je vidět ve výpisu kódu 2. Tímto přístupem lze využít polymorfismus. V případě využití více platforem s různými souborovými systémy se pro každou vytvoří vlastní odvozená třída. Při prvním volání `GetInstance` se pak zvolí správná implementace. A to je další pozitivum tohoto řešení. Instance se vytvoří až když je potřeba. Do té doby instance neexistuje a nezabírá místo v paměti. Tomuto přístupu se říká Odložená inicializace (Lazy Initialization). [2]

1.7 Command

Návrhový vzor Command lze dobře ilustrovat na konkrétním případu užití. V hypotetické hře chceme naprogramovat ovládání hlavního hrdiny. Začněme jednoduchým řešením. Vytvoříme

```
public class Tutorial {
    bool playerHasJumped = false;
    Player player;

    public Tutorial(Player p) {
        player = p;
        player.onJump.AddListener(OnPlayerJump);
    }

    public void OnPlayerJump() {
        playerHasJumped = true;
        p.onJump.RemoveListener(OnPlayerJump);
        // start next lesson
    }
}

public class Player() {
    public UnityEvent onJump = new();

    void HandleInput() {
        if (Input.GetKeyDown(KeyCode.Space)) {
            // make player jump
            onJump.Invoke();
        }
    }
}
```

Výpis kódu 1 Ukázka použití návrhového vzoru Observer

metodu, která zpracovává vstup a při stisku tlačítek na ovladači[3] zavolá správnou akci. Náznak konkrétní implementace lze vidět ve výpisu kódu 3, se kterou se pojí značný problém. Schopnosti hráče jsou v kódu přímo napojeny na určitá tlačítka. Moderní hry ale často umožňují hráči nastavit si vlastní ovládání.

K přidání této funkcionality využijeme návrhový vzor Command. Základní třída `Command` zastřešuje všechny příkazy a má pouze jednu abstraktní metodu `Execute`. Odvozené třídy reprezentují jednotlivé akce pomocí implementace výše zmíněné metody. Každému tlačítku přiřadíme správnou akci, ale protože všechny jsou stejného typu (mají stejnou základní třídu), můžeme je libovolně prohazovat. Ukázková implementace je k vidění ve výpisu kódu 4.

Mezi další přínosy vzoru Command se řadí oddělení akce od objektu a poměrně snadné zajištění „Undo“ a „Redo“. K využití těchto funkcionalit je potřeba implementaci trochu upravit. Pro oddělení akce od objektu musí metoda `Execute()` přijímat jako parametr objekt, se kterým bude manipulovat. Hlavička se tedy změní na `Execute(Actor actor)`. Nyní není předem známo, kdo je cílem. To se může hodit třeba při využívání různých nástrojů ve farmářské hře. Při výběru motyky se na levé tlačítko myši přiřadí instance třídy `TillCommand`. Po kliknutí se tato akce provede na záhon, který se aktuálně nachází pod kurzorem.

Implementace „Undo“ a „Redo“ je ještě trochu složitější. Nejprve do základní třídy `Command` přidáme další metodu `Undo`. Ta provádí akci opačnou k metodě `Execute`. Data, která jsou potřeba pro správnou manipulaci dopředu i dozadu objektem, uložíme do příkazu v jeho konstruktoru. Pro posun šachové figurky je tedy potřeba znát pozici před a po tahu. Nakonec je nutné uložit

[3] Předpokládejme ovladač s tlačítky A, B, X a Y.

```
public class FileSystem {
    public static GetInstance() {
        if (instance == null)
            instance = new FileSystem();
        return instance;
    }

    private static instance;

    // code
}
```

Výpis kódu 2 Jednoduchá implementace návrhového vzoru Singleton

```
void HandleInput() {
    if (IsPressed(BUTTON_X)) Jump();
    else if (IsPressed(BUTTON_Y)) FireGun();
    else if (IsPressed(BUTTON_A)) SwapWeapon();
    else if (IsPressed(BUTTON_B)) Reload();
}
```

Výpis kódu 3 Přímé zpracování vstupu

akce do nějaké kolekce a uchovávat si ukazatel na poslední provedenou. Nyní pomocí `Execute` a `Undo` můžeme provádět akce nebo je rušit a vrátit se do dřívějšího stavu. [2]

```
public class Command {
    public virtual void Execute();
}

public class JumpCommand : Command {
    public override void Execute {
        Jump();
    }
}

public class FireGunCommand : Command {
    public override void Execute {
        FireGun();
    }
}
```

Výpis kódu 4 Přímé zpracování vstupu

Kapitola 2
Analýza současného stavu

Tato kapitola představuje interaktivní stěnu, umístěnou v laboratoři ggLab. Kapitola stručně popisuje fungování interaktivní stěny a identifikuje problémy, které je nutné při vývoji zohlednit. Následuje popis a analýza her, které jsou na interaktivní stěně dostupné k vyzkoušení v době psaní této bakalářské práce. Kapitola je uzavřena důkladnou analýzou hry Spaceship Shenanigans, na kterou tato práce navazuje. Tato analýza zkoumá celkový stav hry i jednotlivé úkoly a vyhodnocuje dotazník, uskutečněný na Dni otevřených dveří.

2.1 Interaktivní stěna a její ovládání

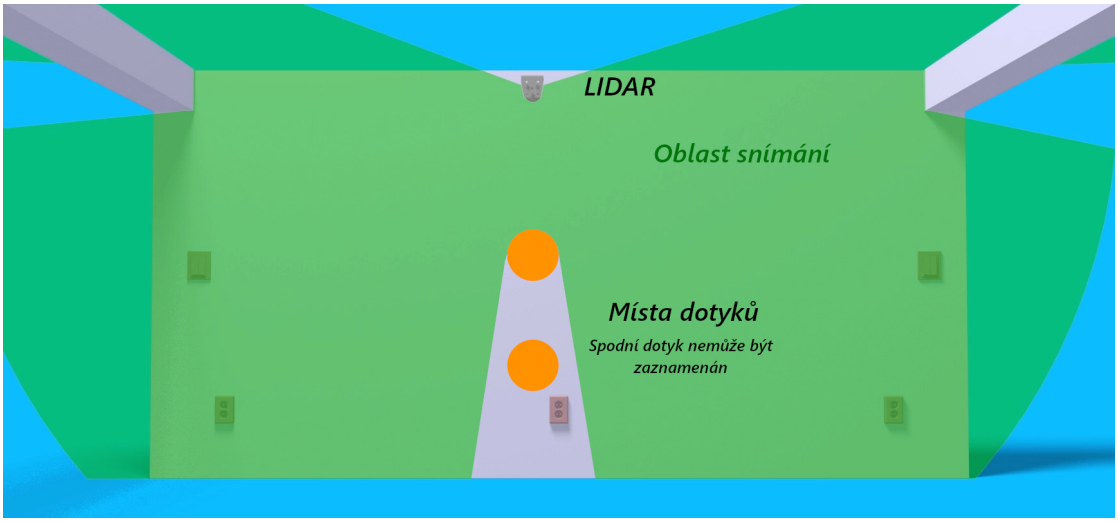
Interaktivní stěna umístěná v grafické laboratoři ggLab byla nainstalována společností INITI Playground. Společně se softwarem určeným ke spouštění aplikací pro ní určených byly dodány i tři hry. Demonz, Save The Planet! a Builders from Mars. Jejich detailnější představení se nachází níže. Samotná interaktivní stěna má celkem jednoduché provedení. Zařízení nevyžaduje instalaci rozměrné zobrazovací, ani dotykové techniky. Na bílou zeď promítá klasický projektor. Snímání dotyku pak zajišťuje LIDAR. Ten pomocí světelných paprsků oskenuje prostor několik centimetrů před stěnou mnohokrát za sekundu. Data ze senzoru jsou poté přenesena do počítače přes síťový kabel s konektory RJ-45. K přenosu se využívá TUIO protokol[1].

Na rozdíl od dotykových obrazovek mobilních telefonů, LIDAR umožňuje ovládat programy promítané na stěně nejen dotykem ruky, ale i jakýmkoliv jiným objektem. Toho se využívá k nezvyklé interakci s počítačem házením molitanových míčků. Tuto aktivitu společnost INITI prezentuje na svých stránkách mezi hlavními přednostmi jejich produktu. Dostat děti k opravdovému hraní od věčného sezení u obrazovek [15].

Při hození se míček ihned odrazí od stěny. Jeho přítomnost je snímána pouze malou chvíli a tak dlouho nezabraňuje paprsky senzoru. Když se ale hráč dotkne stěny rukou, tak blokuje paprsky delší dobu. Senzor v tuto chvíli nemůže snímat dotyky, které jsou z jeho pohledu schované za tím prvním. To lehce omezuje možnosti herního designéra v rozmístění interaktivních prvků. Předpokládejme, že LIDAR je umístěn nahoře ve středu promítaného obrazu. Pokud by byly umístěné dva herní objekty ve středu obrazovky přesně pod sebou, tak kdykoliv by hráč držel vrchní prvek, spodní by nereagoval. Diagram této situace lze vidět na obrázku 2.1.

Dalším problémem je, že senzor snímá objekty přibližně 5 centimetrů od stěny. Při házení míčků se to nijak negativně neprojevuje. Naopak kdyby byla frekvence snímání dostatečně nízká a snímání by probíhalo těsně u stěny, nemusel by senzor míček vždy registrovat. Co se ale dotyku rukou týče, tak vzdálenost snímání dělá tuto interakci neintuitivní. V porovnání s dotykovými obrazovkami, kde uživatel musí fyzicky sáhnout na displej, zde se stačí ke stěně přiblížit. Nejednou

[1] http://www.tuio.org/



Obrázek 2.1 Diagram zastiňování dotyků. Horní objekt blokuje světelné paprsky a senzor tedy nemůže snímat objekt spodní. Zdroj Initi [15], upraveno.

se při používání interaktivní stěny stalo, že volně se pohupující oblečení bylo zaznamenáno jako dotyk, který například spustil aplikaci.

Na umístění samotného projektoru také záleží. U her zaměřených na házení míčkem nemusí nastat problém. Uživatelé se přirozeně vzdálí od stěny, aby mohli lépe vidět celou situaci a napřáhnout se k hodu. Pokud ale hra vyžaduje dotyk rukou, hráč musí pochopitelně být poměrně blízko u stěny. Bude tedy stát ve světle projektoru a stínit si hru. Tento problém lze eliminovat projekcí na krátkou vzdálenost nebo umisťováním prvků výše ve scéně.

2.2 Demonz

Demonz je jedna z trojice her, která byla dodána spolu s interaktivní stěnou. Hra je zasazena ve westernovém městečku, které přepadli roztomilí démoni. Cílem hráčů je tyto démony trefit míčkem, popřípadě se jich dotknout rukou. Když tak udělají, démon zmizí. Ve středu obrazovky je velký starý dům s oknem, hodinami, barelem a jinými tématickými dekoracemi. Po nich démoni skáčou, lezou či na nich sedí. Každý má jednoduchou, ale poutavou animaci. Ta vždy pasuje k pozadí. Například pokud se démon houpe, pak je houpačka někde uchycena, třeba pod větví. Kromě lovení démonů mohou hráči interagovat s některými objekty v pozadí. Okno domu se dá rozbít, z hodin po úderu vypadnou ozubená kolečka a poletující uschlé křoví lze zapálit. Interaktivní prvky obohacují zážitek, ale kromě zajímavé animace či efektu nemají žádný vliv na hratelnost. I přesto byla zábava je v průběhu hraní objevovat.

Demonz má dva režimy. Nekonečný a časovaný. Z názvu prvního vyplývá jeho podstata. Není zde žádný časovač, ani čítač skóre. Neexistuje ani způsob jak hru dohrát, či pokazit. Časovaný režim zvyšuje skóre za každého trefeného démona, přičemž malý čítač je schovaný v horním rohu obrazovky. Také lze nastavit dobu trvání hry. Po uplynutí přiděleného času se mohou hráči zapsat do tabulky skóre a porovnat se s jinými týmy.

2.3 Save The Planet!

Ve hře Save The Planet! se hráči, jak název napovídá, vžijí do role ochránců planety. Ta je pod útokem mimozemských lodí a také na ní padají asteroidy. Tyto objekty mohou hráči zničit

jednoduchým dotykem. Avšak herní pole je plné nepřátel a tak je pořád co dělat. Na pomoc v obraně mají hráči satelity, které mohou aktivovat dotykem. Aktivovaný satelit vystřelí laser, který po dobu několika vteřin zničí vše, čeho se dotkne.

Cílem hry je udržet planetární štít funkční. Jeho životnost je naznačena malým písmem nad planetou v procentech. Obdobně jako u Demonz, samotné herní mechaniky jsou upozaděny a Save The Planet! je více zaměřeno na jednoduchost a vizuální zážitek. I přes to jsem při zkoušení hry měl problém pochopit, co je cílem a jak něj dosáhnout. A když se planetární štít po několika minutách rozbil, nebylo mi jasné, proč tomu tak bylo a jak tomu příště zamezit.

2.4 Builders from Mars

Zlí mimozemští developeři chtějí převzít kontrolu nad městem stavěním panelových domů. Takto je popsaná premisa hry Builders from Mars na oficiálních stránkách.[2] Hráči se snaží tyto stavby rozbít. Domy jsou stavěny po patrech a každé má na sobě nevýrazné zlomené srdce. Kliknutím na něj srdce zčervená a opraví se. Po opravě všech srdcí na patře patro vybouchne a dům se sníží. Hra obsahuje i další interaktivní prvky jako vesmírné lodě snažící se unést krávy. Ty hráči mohou zachránit dotknutím se lodi.

Hráči ale není nijak naznačeno, že by se měl snažit nevýrazná srdce zasáhnout. Cíl hry, bořit budovy, mi také nebyl jasný. Podle vizuální stránky hry jsem spíše očekával, že budovy se mají stavět. Podobně jako u výše zmíněných her je ale stále na co klikat a nelze to udělat špatně.

2.5 Kačky

Hra Kačky je výsledkem semestrální práce tříčlenného týmu[3] v předmětu Softwarový týmový projekt 2. V této hře se hráči snaží trefit kachny své barvy. Ty chaoticky létají po hracím poli a může být obtížné trefit tu svou. Každý hod je riskantní, protože hráč může trefit soupeřovu kachnu a tím mu získat skóre.

Hra je obsahově malá a i přestože působí nedokončeně, je stále zábavné jí hrát. Snažit se trefit poletující kachny je poutavá premisa a házení míčků je zábavné. Na začátku hry lze vybrat počet hráčů, kterých může být až dvanáct. Každý by měl mít možnost si zvolit barvu svých kachen, ale RGB paleta bohužel nefunguje. Tento ovládací prvek stejně není vhodný pro interaktivní stěnu, protože vyžaduje přesnost vstupů, kterou LIDAR nedokáže poskytnout. Výchozí barvy jsou často podobné a v průběhu hraní pak není snadné poznat která kachna patří komu. Zde by byla vhodnější omezená paleta předem vybraných barev.

2.6 Shrnutí

Hry dodané při instalaci interaktivní stěny, tedy Demonz, Save The Planet! a Builders from Mars, jsou velmi jednoduché a zaměřené na děti. Hlavní způsob interakce s těmito hrami je házení míčků na herní prvky. Možnosti akcí jsou omezené pouze na dotknutí se objektu, čímž se mu změní stav (například aktivace laseru) nebo se zničí. Z pohledu herních mechanik jsou tyto hry velmi prázdné. I přes to se v některých hrách nachází trochu komplexnější interakce.

Na první pohled se může zdát jako problém absence tutoriálu či naznačení toho, co má hráč dělat. Avšak tyto hry jsou cílené na děti a jsou především umisťované do veřejných prostor. Výhoda této cílové skupiny je, že typicky někdo danou hru bude již hrát. Stačí jen chvíle pozorování někoho jiného a pravidla jsou ihned jasná. Dále je možné hry pochopit pouze metodou pokus-omyl. Díky tomu, že hráč nemůže udělat nic, co by bylo penalizováno, umožňují tyto hry experimentaci. Stačí se dotýkat různých objektů a pozorovat, co se stane.

[2] https://www.initiplayground.com/games

[3] Autoři: Adam Štursa, Kateřina Hrdličková a Viktor Pašek

Obrázek 2.2 Přehled her pro interaktivní stěnu. Snímky obrazovky pořízené přímo z her. Builders from Mars převzato z webu Initi [15].

2.7 Spaceship Shenanigans - přehled

Na Spaceship Shenanigans přímo navazuje tato bakalářská práce a proto se jí ze všech her zde věnuji nejvíce. Tato hra byla vytvořena jako semestrální práce v předmětu Softwarový týmový projekt 1 (BI-SP1) týmem šesti studentů, včetně mě[4]. Výsledek je kompletní hra, kde lze soutěžit s jinými týmy o nejvyšší skóre v žebříčku.

V této kooperativní hře se tým hráčů ocitne v roli posádky vesmírné lodi. Avšak jejich plavidlo potřebuje neutuchající údržbu a opravy. Tyto úkoly se hráčům opakovaně objevují na obrazovce a každý má časový limit, ve kterém je nutné ho splnit. Za vyřešení úkolu jsou hráči odměněni určitým počtem bodů skóre. Naopak při neúspěchu je tým penalizován ztrátou života. Tato herní smyčka se opakuje tak dlouho, dokud loď ještě nějaké zdraví má. Po ztrátě posledního života se hra ukončí a hráčům se ukáže tabulka nejvyšších skóre. Pokud hraním dosáhli dostatečného počtu bodů, tak se do tabulky umístí. Ještě před tím ale musí zadat jméno týmu.

Jak již bylo zmíněno, základní herní smyčka se skládá z plnění úkolů. Jejich náplň je jednoduchá a vyřešení obvykle netrvá více než deset vteřin. Samozřejmě za předpokladu že hráč neudělá chybu. Ta není nijak penalizována a úkol je možné zkoušet dokud nevyprší časový limit. Zápletkou je tedy fakt, že na obrazovce se v podstatě vždy nachází více úkolů najednou. Hráč si tedy vybírá úkol u kterého zbývá nejméně času. Díky tomu typicky hráč nezačíná řešit úkol na jeho začátku, ale až po uplynutí nějaké doby. Skutečnost, že čas na splnění utíká, donutí hráče chvátat a tím se zvýší šance chyby. Na hráče také působí lišta časovače. Ta mění barvu v závislosti na zbývajícím čase. Nejprve je bílá, poté žlutá a v posledních chvílích červená.

Různé úkoly mají různou náročnost a proto jsou odlišně nastaveny. S vyšší složitostí přichází jak více času na splnění, tak větší odměna za úspěch. Snazší úkoly to mají naopak. Další dělení je na dotykové a házecí. Dotykové úkoly se nacházejí v nižší a střední části obrazovky. Jsou navržené pro ovládání dotykem ruky a mohou si dovolit komplexnější vstupní metody, jako je klávesnice. Často vyžadují více kliknutí než úkoly házecí. Ty se naopak objevují ve zbývající horní části. Pro přizpůsobení ovládání míčkem jsou typicky snazší a testují přesnost hodu hráče.

[4]Autoři: Matěj Chlan, Magdaléna Musilová, Zdeněk Nejedlý, Šimon Taněv, Bui Tomáš Pham a Viktor Pašek

Mezi kladné stránky hry řadím jasnou komunikaci hráči, kam má upoutat svou pozornost. Úkoly jsou ohraničené, aby byl jasně naznačen jejich rozsah a aby se skryly nedokonalé kraje. Lišta časovače, která je součástí ohraničení, na první pohled předá hráči velmi důležitou informaci několika způsoby. Z délky lišty lze jednoduše poznat kolik poměrově zbývá času. To se pojí s rychlostí jejího růstu, ze které hráč odhadne, kolik ho zbývá celkově. Nakonec barva lišty hráče ujistí o závažnosti problému. V pozadí se pouze pohupuje loď plující vesmírem. Ta je záměrně nevýrazná, aby úkoly byly oproti ní kontrastní. Hráč se tedy může zaměřovat pouze na to důležité.

Jako neblahou hodnotím komunikaci k hráči ohledně stavu hry. Uživatelské rozhraní, které je poněkud nenápadné, hráč může jednoduše ignorovat, či si ho vůbec nevšimnout. Problém je to především u čítače životů. Ten je v levém horním rohu, kam se hráči obvykle nedívají. Stává se pak, že lodi dojdou všechny životy a hra skončí, aniž by hráči tušili, že se tato událost blíží. Nikde také není naznačeno, že byl ztracen život. Po vypršení času úkol pouze zmizí a změní se číslo v uživatelském rozhraní. Tento problém se projevuje nejvíce, pokud hráč řeší úkol těsně před vypršením jeho limitu. Pak se může stát, že hráč neví, jestli úkol dokončil nebo nestihl. Z čítače životů tuto informaci také nevyčte, pokud se na něj nepodíval chvíli před tím. I tak život mohl odebrat jiný úkol. Čítač skóre má podobné problémy, ten ale pro průběh hry není tak důležitý.

Další funkcí, která chybí Spaceship Shenanigans, je absence jakékoliv změny obtížnosti. Počet a délka úkolů je stejná jak pro samotného hráče, tak pro početný tým lidí. Poté ve průběhu hry se tyto počty také nemění a obtížnost zůstává stejná. Pokud se hráči naučí dobře a efektivně plnit úkoly, tak mohou hrát prakticky do nekonečna. A přesně to se při testování stalo. Jediným důvodem pro ukončení byla repetetivnost a ztráta jakékoliv překážky k překonání.

2.8 Spaceship Shenanigans - úkoly

Konečně se dostáváme k neustále zmiňovaným úkolům. V této kapitole budou postupně představeno a podrobně rozebráno všech devět úkolů. Jmenovitě Tlačítko, Kód, Žárovky, Jeřáb, Pumpování, Výměna baterky, Pojistky, Společná tlačítka a Basketbal.

Úkoly se objevují na pěti místech. Každé místo může vytvořit několik druhů úkolů. Naopak některé úkoly mohou být pouze na jednom místě. Například Pumpování je vždy na zádi lodi a Pojistky na přídi. Úkoly se dělí na dva druhy, dotykové a házecí. Jak již bylo zmíněno, dotykové úkoly se nacházejí níže a měly by se řešit pomocí dotyku rukou. Házecí úkoly se objevují pouze vysoko na stěně, aby hráči byli motivováni využívat míčky.

2.8.1 Tlačítko

Začneme nejjednodušším úkolem ve hře. Jediný objekt v tomto úkolu je tlačítko, které potřebuje zmáčknout. Výhoda této jednoduchosti je odpočinkovost. V ideálním případě by hráč měl být ponořený do hry a pořád něco dělat. To může způsobit lehce hektickou situaci. Hráč si může u tohoto úkolu trochu oddechnout a přehodnotit stav hry. Ovšem tato triviálnost může být dvojsečná zbraň. Pokud hráč nestíhá, může se rozhodnout přeskočit tento jednoduchý úkol a začít řešit nějaký náročnější. Pak se může stát, že na tlačítko zapomene a zbytečně připraví svůj tým o život.

K tomuto úkolu se váže i zasloužená kritika. Vzhledem k tomu, že jde opravdu pouze o stisknutí tlačítka bez jakéhokoliv kontextu, tak při testovaní někteří hráči nechápali, proč hra takový úkol vůbec má.

2.8.2 Kód

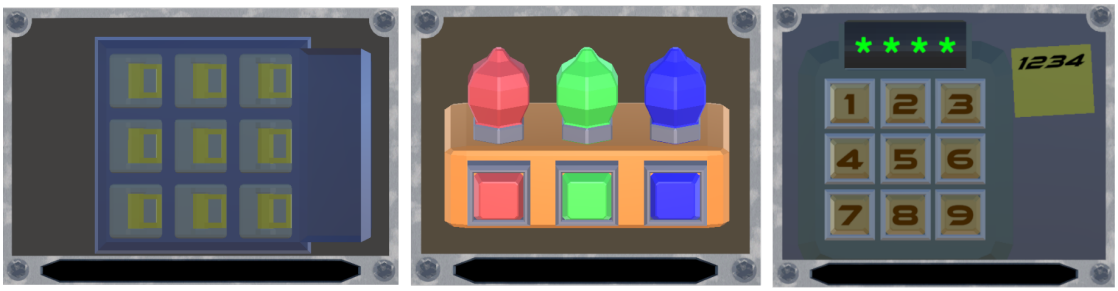
Tento velmi intuitivní úkol vyžaduje po hráči zadat určenou sekvenci čísel na klávesnici. Požadovaný čtyřmístný kód lze nalézt na papírku přilepeném na straně zařízení. Pokud hráč při zadávání udělá chybu, pak se zadaná část kódu smaže a musí začít znovu.

Klávesnice, na které hráč zadává kód, je velmi problémová. Jádrem potíží jsou tlačítka. Ta jsou implementována jako samostatné vstupy, které mezi sebou nemohou komunikovat. Kód dodaný od autorů zařízení registruje dotyk jako kruh s určitým poloměrem kolem bodu. Jeden dotek se pak může projevit jako zmáčknutí dvou či více tlačítek, což vede k chybě a vymazání kódu. Tento problém se projevil až při testování na stěně, protože kliknutí myší v editoru je bráno jako bod. Na opravení problému nezbyl v semestru čas.

2.8.3 Žárovky

Žárovky jsou inspirované v celku populární hádankou obvykle nazývanou „Simon says“. Cílem je zopakovat sekvenci bliknutí pomocí stisků příslušných tlačítek. V této verzi jsou tři žárovky, červená, zelená a modrá. Pod nimi jsou umístěná příslušná tlačítka stejných barev. Kód má vždy délku čtyř bliknutí.

Testování ukázalo, že hráči mají s tímto úkolem problém. A ne jejich vinou. Zaprvé, jediná zpětná vazba, kromě animace promáčknutí tlačítka, je po stisku špatného tlačítka. To se projeví červeným zábleskem přes celý úkol. Hráči by měl být indikován ale i správný postup, například řadou LED diod. V současné podobě tedy není jasné, jestli hráč postupuje dobře. Zadruhé jsou tlačítka moc blízko u sebe a vzniká podobný problém jako na klávesnici v úkolu Kód s mačkáním jich více najednou. Zatřetí, stěna snímá dotek o několik centimetrů dřív. V případě že hráč chce rychle stisknout levé a pak pravé tlačítko, omylem se dotkne i prostředního, protože svou ruku nevzdálil dostatečně daleko od stěny.



Obrázek 2.3 Méně oblíbené úkoly. Zleva Pojistky, Žárovky a Kód.

2.8.4 Jeřáb

Jeřáb je oblíbený úkol připomínající pouťové automaty na plyšáky. Ve vitríně je umístěný objekt, který je potřeba zvednout. Nad ním se zleva doprava pohybuje jeřáb. Po stisku tlačítka jeřáb sjede dolů a pokusí se objekt uchopit. Pokud se jeřáb nacházel přesně nad ním, objekt zvedne a úkol bude splněn.

Tento úkol vyžaduje dobré načasování. Pozice jeřábu nemusí být úplně přesná a pozorný hráč si může všimnout malého zarovnání po stisku tlačítka v dostatečně dobré pozici. Nejen díky tomu patří mezi oblíbené úkoly.

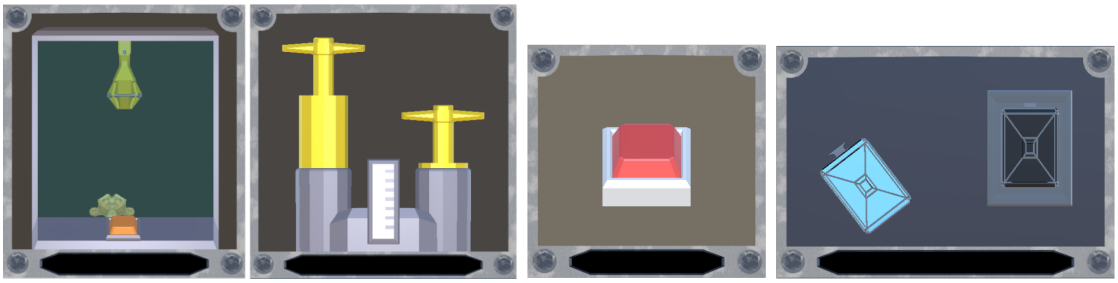
2.8.5 Pumpování

Zde se před hráčem objeví pumpa se dvěma stlačitelnými částmi a ukazatelem tlaku mezi nimi. Smyslem je ťuknout na vysunutou část, která se poté zasune a zároveň vysune druhou část. Dále hráč klikne na druhou, nyní vysunutou, část a proces proběhne naopak. Takto hráč postupuje, dokud není ukazatel mezi trubicemi plný.

Pumpování je oblíbené díky jeho rytmičnosti. Když hráč najde správné tempo, stane se tento úkol velmi zábavným a rychlým. Ukázal se ale jeden problém. Někteří uživatelé chtěli za pumpu chytit a stáhnout jí dolů ručně. Nikdy ale netrvalo dlouho zjistit, že se stačí dotknout.

2.8.6 Výměna baterky

Tento úkol velmi špatně komunikuje s hráčem. Cílem je vyměnit starou baterku za novou postupným klikáním na zmíněné baterky. Problém je v tom, že baterky nevypadají jako baterky a hráči tedy nechápu, co se děje. Nikde není ani naznačeno, že jedna baterka je vybitá a druhá nabitá. Když bylo hráčům řečeno, že mají vyměnit baterky, tak se obdobně jako u Pumpování snažili baterku chytnout a přetáhnout ručně. Avšak baterka reaguje na pouhé kliknutí.



Obrázek 2.4 Oblíbenější úkoly. Zleva Jeřáb, Pumpování, Tlačítko, Výměna baterky.

2.8.7 Pojistky

Cílem tohoto úkolu je vypnout všechny pojistky, které se nachází ve skříncе. Vypnutí se provede jednoduchým kliknutím na prvek. Pojistek je celkem devět a po vypnutí je už není možné zapnout. To se zavedlo kvůli tomu, že hráči není nijak naznačeno, co má dělat a jaký stav je správný. Tedy nemůže udělat chybu. Animace vypínání pojistek je zleva doprava a ne shora dolů, jak je tomu v reálných zařízeních a to může hráče zmást. Tento úkol má obecně problém jasně říct hráči, co vůbec znamená a co je jeho cílem.

2.8.8 Společná tlačítka

Konečně se dostáváme k úkolu navrženém pro házení míčkem. Na stěně se objeví dvě tlačítka a pokud je jedno zasažené, pak se na něm spustí třívteřinový odpočet. Hráči mají tedy trefit obě dvě tlačítka v rozmezí maximálně tří vteřin od sebe. Tento úkol podporuje spolupráci, protože je jednodušší hodit oba míčky zároveň společně s dalším hráčem. I zde je ale problém komunikace. Není nijak naznačeno, že mají být stisknuta obě tlačítka a hráči to zjistí spíše metodou pokus omyl.

2.8.9 Basketbal

Jako poslední úkol zbývá Basketbal. Vysoko na stěně se ukáže basketbalový koš. Tam, kam se hráč trefí míčkem se objeví basketbalový míč. To je jediný objekt ve hře, na který působí gravitace a míč buď spadne do koše nebo se odrazí mimo něj. Tento úkol je mezi hráči jeden z nejoblíbenějších. Je krásně čitelný a na první pohled bylo každému jasné, co má dělat. Zároveň si hráči více užívají házení míčků než dotýkání se stěny. Za zmínku stojí fakt, že míče mohou vypadnout mimo ohraničení úkolu. To ale v průběhu testování nikdy nezpůsobilo problém a tak tato chyba zůstala neřešená.

Obrázek 2.5 Úkoly vyžadující házení míčků. Vlevo Basketbal, vpravo Společná tlačítka.

2.8.10 Obecná pozorování

Mnoho úkolů v této hře trpí špatnou komunikací k hráči. Pokud jim není ihned jasné co mají dělat nebo s čím mohou interagovat, tak po několika neúspěšných dotycích ztratí zájem a zmateni zkusí něco jiného. To obvykle vede k vypršení časovače a ztrátě života. Když ale bylo hráčům ukázáno, jak se má úkol plnit, tak byl jejich zájem o hraní mnohem vyšší.

Další problém mnoha úkolů je samotné ovládání. Senzor snímá dotyk přibližně pět centimetrů od stěny. Hra tedy reaguje dříve než se jí hráč dotkne. To je pro mnoho lidí velmi neintuitivní. Příkladem buď výše zmíněná klávesnice v úkolu Kód. I když hráč trefí pouze jedno tlačítko, musí pak zvednout ruku dostatečně daleko od stěny, aby ho přestala snímat. Pak teprve může pokračovat dalším kliknutím. Ale při testování lidé obvykle nadzvedli prst pouze trochu tak, aby se nedotýkali stěny. Tento problém se projevuje pouze při dotyku. Míček se po dopadu vždy odrazí a vyvolá pouze jedno kliknutí.

2.9 Dotazník

Na akci Den otevřených dveří uskutečněné v listopadu 2023 si návštěvníci mohli vyzkoušet Spaceship Shenanigans, společně s mnoha dalšími hrami, v laboratoři ggLab. Po dohrání byli požádáni o vyplnění krátkého papírového dotazníku. Ten se skládá ze čtyř otevřených otázek a dvou uzavřených. Na otazník celkem odpovědělo 69 respondentů.

1. **Bavila Vás hra? Co se líbilo a co naopak ne?**

   V této první otázce mohli respondenti vyjádřit svůj celkový názor na hru. Odpověď byla otevřená a tak měli šanci napsat o tom, co je nejvíc zaujalo ať už v pozitivním, či negativním smyslu. Většina odpovědí (34 odpovědí, 49%) hodnotila hru kladně neobsahovala konkrétní pochvaly či kritiku. 7 respondentů (10%) chválilo využití míčků nebo úkoly, které je využívaly a tři odpovědi (4%) chválily specificky úkol Basketbal.

   Hlavním bodem kritiky v této otázce byly úkoly Žárovky a Kód (4 odpovědi, 6% a 3 odpovědi, 4% respektive). Dva respondenti (3%) sdělili, že by úkoly mohly měnit pozice, kde se objevují. Ve výsledku většina účastníků hodnotila celkový dojem ze hry kladně.

2. **Jak hodnotíte ovládání hry? Měl/a jste s něčím určitým problém?**

   Druhá otázka byla také otevřená. Hráči hodnotili ovládání hry a jeho obtíže. Hlavní část odpovědí byla tvořena strohými pochvalami. Odpovědí typu „Bez problémů“, „Líbí se mi“ nebo „ok“ v dotazníku bylo 23 (33%). Obdobně jako v první otázce i zde největší kritiku obdržely úkoly Kód (17 odpovědí, 25%) a Žárovky (5 odpovědí, 7%). Pouze se prohodilo jejich pořadí v nespokojenosti.

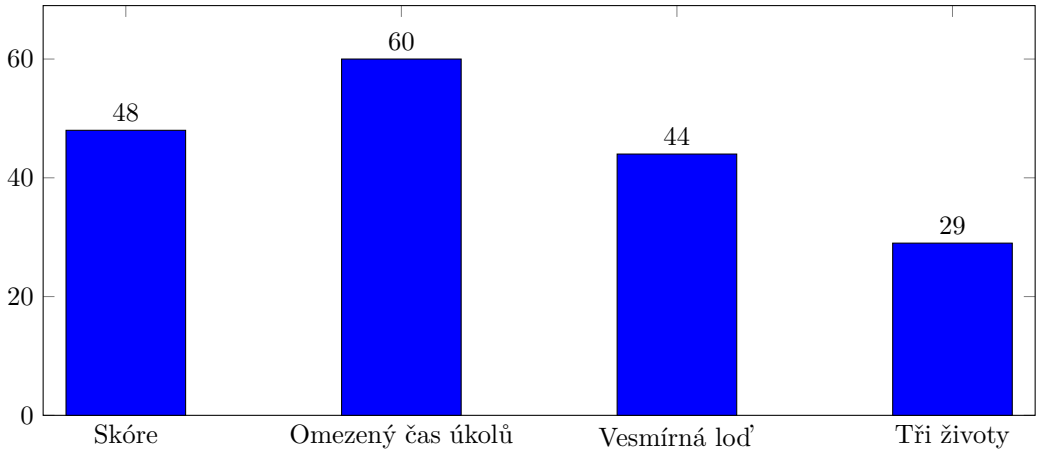
   Míčky nebyly zdrojem pouhých pochval. Dva respondenti sdělili, že obtíže jim dělalo chytání míčků, potažmo jejich kutálení. Zcela nepřekvapivě se třem respondentům nelíbila vzdálenost

snímání. Stěna registruje dotyk už ve vzdálenosti několika centimetrů a to často působí potíže. Tato skutečnost musela být frustrovaným hráčům vysvětlována.

3. Zaškrtněte z následujícího prosím to, čeho jste si ve hře všiml/a.

Tato otázka s uzavřenou odpovědí obsahuje čtyři tvrzení popisující vlastnosti Spaceship Shenanigans. „Hra se odehrává na letící vesmírné lodi.“, „Ve hře je skóre, které se na konci hry ukládá do tabulky nejlepších.“, „Hra končí ve chvíli, kdy hráči nestihnout splnit tři úkoly.“ a „Na každý úkol, který se objeví, je omezený čas.“. Respondenti měli zaškrtnout ty vlastnosti, kterých si všimli.

Převážná většina účastníků zaznamenala omezený čas na úkoly. Čas se zobrazuje na ohraničení každého úkolu a tedy takový výsledek není překvapivý. Nadpoloviční část respondentů si všimla, že hra obsahuje skóre. Dotazník ale nerozlišoval jestli si ho všimli v průběhu hry, nebo až po jejím skončení. Lze usoudit, že díky zobrazení tabulky nejvyšších skóre po konci každé hry, je o něm vyšší povědomí než o životech. Těch si všimla necelá polovina hráčů. Přesné hodnoty lze vidět na obrázku 2.6.



Obrázek 2.6 Graf ukazující, kolik hráčů si všimlo jednotlivých částí hry.

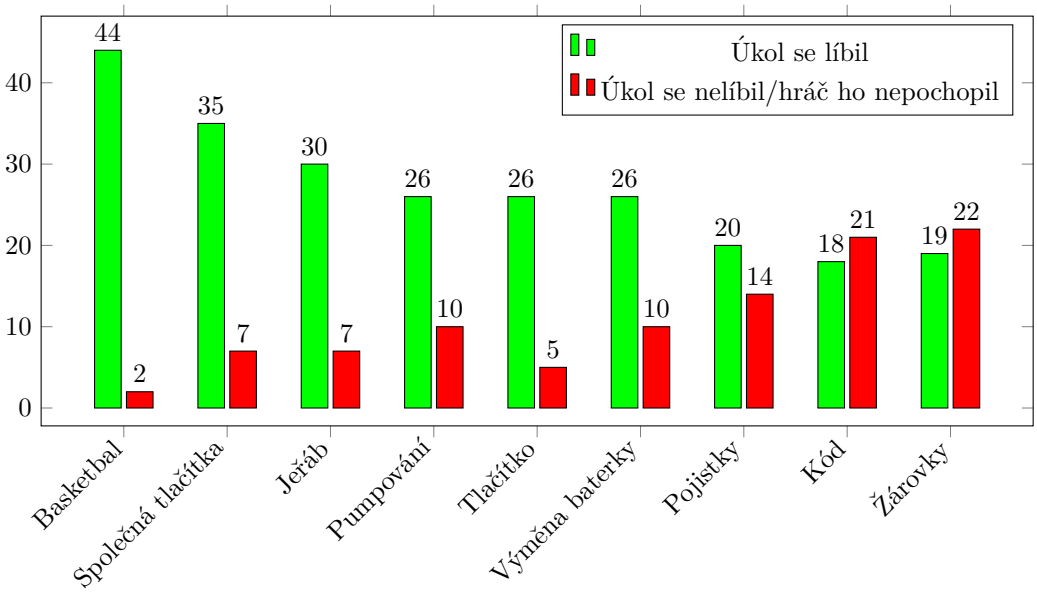
4. Hodnocení úkolů.

Otázka číslo čtyři zjišťovala oblíbenost úkolů. Respondenti měli zakroužkovat úkoly, které se jim líbily a škrtnout ty, které neplnili rádi nebo je nepochopili. Graf oblíbenosti je vidět na obrázku 2.7. Ve shodě s předchozími otázkami se nejoblíbenějším úkolem stal Basketbal. S úkolem Společná tlačítka na druhém místě je vidět, že úkoly využívající míčky jsou preferované. Jako třetí až sedmý se umístily úkoly s jednoduchým ovládáním. Kód a Žárovky jsou nepřekvapivě hodnoceny spíše záporně. Tomu je nejspíše díky jejich proradnému ovládání.

5. U interaktivní zdi může hrát více hráčů zároveň. Měl/a jste při hře nějaké spoluhráče? Kolik? Jaké to bylo?

V této otázce bylo zjišťován obvyklý počet hráčů a popřípadě hodnocení jejich spolupráce. Díky otevřenosti odpovědi ale některé hodnoty byly nepřesné (odpovědi typu „mnoho“ nebo „3-4“). Odpovědi na hodnocení spolupráce mohly znamenat buď kolegialitu spoluhráčů či vůbec hodnocení toho, že hra je kooperační.

Šest respondentů uvedlo, že hru hráli sami. Ve většině případů byl tedy u stěny tým lidí. 22 odpovědí tvrdí, že měli jednoho až dva spoluhráče a 24 respondentů jich mělo tři a více. Velikost týmu tedy byla nejčastěji tři až čtyři, což odpovídá i velikosti skupin zájemců o studium. Příjemné bylo zjištění, že většina hráčů hodnotila spoluhráče kladně. Poměr odpovědí oceňujících spoluhráče ku nespokojenosti s nimi byl 21:2.



Obrázek 2.7 Graf oblíbenosti úkolů.

6. **Další poznámky/nápady na další hru.**

   V poslední otázce bylo hráčům umožněno vyjádřit se o čemkoliv dalším ve hře a popřípadě navrhnout téma nějaké hry, kterou by rádi viděli na stěně. Mezi odpověďmi lze najít výtku k uživatelskému rozhraní, či návrhy k dalším úkolům. Někteří respondenti by chtěli větší míru spolupráce, či dokonce úkoly, které na sobě závisí. Nápady na nové hry se dělí do tří kategorií. Sportovní hry, jako tenis nebo volejbal, existující hry upravené pro stěnu, jako Space Invaders nebo plošinová skákačka a „Versus“, kde budou hráči postaveni proti sobě.

2.10 Kód Spaceship Shenanigans

Hra Spaceship Shenanigans byla vytvořena jako semestrální práce v předmětu Softwarový týmový projekt 1 v akademickém roce 2022/2023. Pracovalo na ní šest studentů včetně mě, autora této bakalářské práce. Má role v týmu byla hlavní programátor s podporou Magdalény Musilové. Díky tomu jsem s kódem velmi dobře obeznámen. Od té doby jsem spolupracoval na další hře[5], ze které jsem nabyl dalších zkušeností. V následující sekci jsem analyzoval Spaceship Shenanigans z hlediska kódu a tyto znalosti jsem využil při implementaci Vesmírných Vylomenin.

Hra byla vytvořena v herním enginu Unity pomocí programovacího jazyka C#. Kód pro komunikaci s interaktivní stěnou nám poskytl Ondřej Průcha, spoluzakladatel INITI Interactive. Dotyky na stěně jsou abstrahovány za třídou `BaseHittable`. Pro vytvoření jakéhokoliv interaktivního prvku je nutné podědit tuto třídu a překrýt (override) metodu `Hit(Vector2 hitPosition)`[6]. Ta je volána pokaždé, když se uživatel dotýká daného objektu a její parametr specifikuje přesné místo zásahu. Oblast pro dotyk je definována komponenty[7] typu `Collider2D`, ne vizuální reprezentací objektu.

LIDAR provede sken celé stěny několikrát za sekundu a pokaždé zavolá zmíněnou metodu `Hit`. Takto získaná informace ale nebyla dostačující a proto byla vytvořena třída `BaseInteractive`.

[5] Kryštof, baron Branický. Hra vytvořená jako semestrální práce předmětu BI-VHS.

[6] `Vector2` je struktura obsahující souřadnice x a y, typicky reprezentující pozici v dvourozměrném kartézském souřadnicovém systému.

[7] https://docs.unity3d.com/Manual/Components.html

Ta zásahy dělí na dotyk, držení a uvolnění, respektive na volání metod `OnPress()`, `OnHold()` a `OnRelease()`. Metoda `OnPress()` je volána při první registraci dotyku nebo pokud nebyl určitou dobu dotyk registrován. Metoda `OnHold()` je volána v ostatních případech. Nakonec zavolání `OnRelease()` je testováno s jistým odstupem času po každém `Hit()`.

Z této třídy jsou odvozené nejen ovládací prvky, jako `GenericButton` nebo `Battery`, ale i všechny úkoly (v kódu nazývané „Task“) přes jejich základní třídu `BaseTask`. Ta má za práci zpracovávat úkony sdílené přes všechny úkoly. Především tedy ukončuje úkol po určité době a ovládá jeho ohraničení zobrazující lištu časovače.

`BaseTask` je zbytečně navázaná na ohraničení. Její kód jak zahajuje, tak pozastavuje vizualizaci časovače. Tím porušuje „Single responsibility principle“. Třída `BaseTask` je díky dědění vstupem. Tohoto vstupu ale ne všechny úkoly využívají. Například `BasketballTask` využívá metodu `OnPress` pro vytváření míčů, ale `ButtonTask` této možnosti nevyužívá. Tato relace také nedává smysl. Úkol není vstup, úkol má vstupy zpracovávat. A v případě, že je potřeba více různých vstupů, nelze dědění z `BaseInteractive` ani využít.

Porušení „Single responsibility principle“ je vidět v mnoha úkolech. `BasketballTask` tvoří míče, `CraneTask` (Jeřáb) hýbe se zvedacím zařízením a `SimonSaysTask` (Žárovky) zobrazuje kód blikáním žárovek. Některé úkoly jsou ale navrhnuty dobře. Například `SwitchTask` (Pojistky) pouze reaguje na vstupy z jednotlivých pojistek. Při změně otestuje, zda jsou všechny zapnuté a v případě že ano, ukončí úkol. Tento a podobně navržené úkoly nedělají jinou práci, než že v reakci na vstupy mění svůj stav. Předat informaci o změně pak mohou předat pomocí návrhového vzoru Observer. Ostatní části úkolu, typicky zobrazování uživateli, zbydou na jiné objekty. Ty mohou nabýt roli pozorovatele.

`GameController` je třída zaopatřující celý průběh hry a je implementována jako Singleton, ať je k ní jednoduchý přístup odkudkoliv z kódu. Uchovává v sobě stav hry, vytváří úkoly a drží globální hodnoty. Zmíněné funkcionality zaopatřují samostatné třídy, které jsou ale definované uvnitř `GameController`. Toto rozhodnutí snižuje čitelnost kódu a vhodnější by bylo je rozdělit do samostatných souborů.

`SpawningController` periodicky vytváří úkoly. Má přehled o všech pěti bodech, kde se mohou objevit a jestli je na nich úkol umístěn. `GameState` je jednoduchá třída ve které je uložen stav hry. Tedy maximální počet životů, současný počet životů a skóre. Nakonec třída `Globals` má poskytovat globální hodnoty. Využitá je ale jen pro výchozí čas na splnění úkolu.

Kapitola 3
Game design dokument

Pro lepší spolupráci v rámci týmu se před začátkem vývoje nebo současně s ním vytváří game design dokument (GDD). Struktura tohoto dokumentu není pevně dána a každý tým, popřípadě vydavatel, si jí určuje sám. Součástí GDD může být například stručné shrnutí hry, popis postav, zápletka, popis náplně hry, návrh vizuálního stylu nebo diagram průběhu hry [16]. GDD je vhodné napsat i pokud na hře pracuje jen jeden člověk. Tento dokument může pomoci udržet vizi hry, určovat, čím má hra být a naopak čím být určitě nemá. Kvalitně napsaný GDD pomáhá rozlišit náročnost jednotlivých částí a díky tomu jim přiřadit odpovídající prostředky [17].

Já jsem se rozhodl k mé bakalářské práci vytvořit GDD z důvodu lepšího představení hry čtenáři. Tento dokument se v průběhu vývoje měnil a nyní zachycuje finální stav hry. Zároveň slouží jako podklad pro softwarový design.

3.1 Náplň hry

Tato práce navazuje na již vytvořenou hru Spaceship Shenanigans. Té se věnuji více dopodrobna v sekci 2.7. Základ hry se proto nemění. Tým hráčů se vžije do role posádky vesmírné lodi, která jako každý pořádně komplikovaný stroj, potřebuje neustálou údržbu, opravy, kalibrace a další úkony. Nejdůležitější činnost, kterou budou hráči dělat, je plnění úkolů. Ty jsou povinné a za jejich splnění je odměna ve formě skóre. Úkoly mohou zahrnovat plno činností, například nastavování souřadnic, či opravy motoru. O úkolech více níže. Dále hráči mohou věnovat svou pozornost jednotlivým místnostem lodi. Každá místnost je interaktivní a po kliknutí na ní se přehraje krátká animace.

Vesmírná loď má svou integritu. Pokud se hráčům nepodaří dokončit úkol v časovém limitu, pak se tato integrita sníží. V případě, že integrita klesne na nulu, se loď zničí a hra skončí. Poté hráči dostanou možnost se podívat do tabulky nejvyšších skóre, kam se mohou i umístit.

3.2 Změny oproti Spaceship Shenanigans

Původní Spaceship Shenanigans mají pevné nastavení poměru stran obrazovky. To stačí pro současnou sestavu v laboratoři ggLab. Do budoucna ale jsou plány stěnu rozšířit a poté by se hra nepřizpůsobila. Problémy bude také dělat její spuštění na stěnách v jiných prostorech, které nemusejí mít poměř 16:9. Proto se Vesmírné vylomeniny přizpůsobují rozměrům stěn. Všechny ilustrace jsou připravené pro rozlišení 4K v základním nastavení velikosti kamery.

Pro různé poměry stran je přizpůsobené i vytváření úkolů. Původní verze má pět statických bodů, na kterých se objevují úkoly. Nově je za běhu vytvořená mřížka, která se přizpůsobuje velikosti obrazovky. Každá buňka této mřížky je místo, kde se může objevit úkol. Jejich počet

je omezený v závislosti na počtu hráčů. Mřížka je rozdělená na dolní a horní řady. Ve spodních řadách se objevují úkoly určené k řešení dotykem ruky. Naopak horní řady, kam hráči nemají dosáhnout, vyžadují ke splnění úkolů využít házení míčkem.

Při navrhování Spaceship Shenanigans bylo rozhodnuto o využití 3D modelů pro reprezentaci objektů ve hře. Až později ale bylo zjištěno, že dodaný balíček pro registrování dotyků nepodporuje perspektivní typ kamery. Při využití ortografického režimu ale objekty natočené přímo ke kameře vypadají ploché. O co hůř, pohyb v ose kamery se vizuálně nijak neprojeví. Animace stisknutí tlačítka tedy nebyla vůbec vidět. To vyústilo v různě nepřirozeně natočené objekty. Vesmírné vylomeniny tedy využívají 2D grafiku.

V původní verzi je loď pouze na okrasu. Je to nevyužitý potenciál. Nově se loď skládá z místností, které reagují na dotyk krátkou animací. Místnosti jsou náhodně generované a jejich počet se přizpůsobuje šířce obrazovky.

Jak bylo zmíněno v analýze, hra nezvyšovala obtížnost a po nějaké době bylo možné se úkoly naučit plnit tak efektivně, že ze hry vymizela jakákoliv překážka. Vesmírné vylomeniny obsahují úroveň nebezpečí. Ta se periodicky zvyšuje. S každým novým stupněm se krátí čas určený k řešení úkolů. Společně s tím se ale zvyšuje násobič skóre. Tedy v pozdější fázi hry je odměna za splnění úkolu násobně vyšší.

Další viditelnou změnou je výběr jazyka. Spaceship Shenanigans má uživatelské rozhraní v angličtině. Vesmírné vylomeniny jsou naopak celé v češtině. Toto rozhodnutí bylo učiněno na základě cílové skupiny. Hra bude prezentována především na fakultě a tak je český jazyk vhodnější.

3.3 Ovládací prvky

Hra má několik typů ovládacích prvků, které se vyskytují napříč úkoly. Jejich opakované využití pomáhá hráčům snáze pochopit možnosti interakcí právě řešeného problému. Tyto prvky jsou základními kameny a jejich různými kombinacemi vznikají zajímavé aktivity.

Tlačítko je nejjednodušší interaktivní prvek. Při dotyku se zmáčkne a stisknuté zůstane, dokud má na něm hráč položenou ruku. Tlačítko je vhodné i pro úkoly, které díky svému umístění vysoko na stěně vyžadují využití míčku. Díky jednoduchosti může být využito v mnoha případech. Typicky jako potvrzovací akce.

Časované tlačítko se předchozímu dost podobá. Také reaguje na dotyk, ale pak vydrží stisknuté jistou dobu. Po jejím uplynutí se uvolní a hráč ho může stisknout znovu. Tento ovládací prvek lze využít i v úkolech vyžadujících míčky.

Držené tlačítko se na první pohled neliší od klasického tlačítka. Ale místo toho, aby výstup byl stisknuté/uvolněné, tento prvek měří délku držení. Při dotyku se spustí časovač, který běží až do uvolnění. Právě kvůli potřebě držení tlačítka, není vhodné ho využít k ovládání míčkem.

Posuvník je sofistikovanější prvek, který má madlo. To se může posouvat po přímé čáře a na této cestě má nastavitelný počet pozic, do kterých po uvolnění zapadne. Minimální počet poloh jsou dvě, jedna na začátku a druhá na konci cesty. Aby hráč mohl s posuvníkem hnout, musí první dotek být v blízkosti madla. Pokud se dotkne příliš daleko, interaktivní prvek jej bude ignorovat.

Mřížka může představovat klávesnici s určitým počtem řad a sloupců kláves. Jde ale vymyslet i neobvyklejší využití. Podobný ovládací prvek lze sestavit z mnoha samostatných tlačítek. Nevýhoda tohoto přístupu ale je, že hráč jich může stisknout více najednou a díky tomu pokazit právě prováděnou akci. Mřížka toto řeší tak, že celá klávesnice je jeden prvek. Při dotyku na

kterémkoliv místě stiskne správné tlačítko, to drží stisknuté a ignoruje ostatní dotyky. Až když se hráč úplně přestane dotýkat části klávesnice, tak se stisknuté tlačítko uvolní a v interakci lze pokračovat.

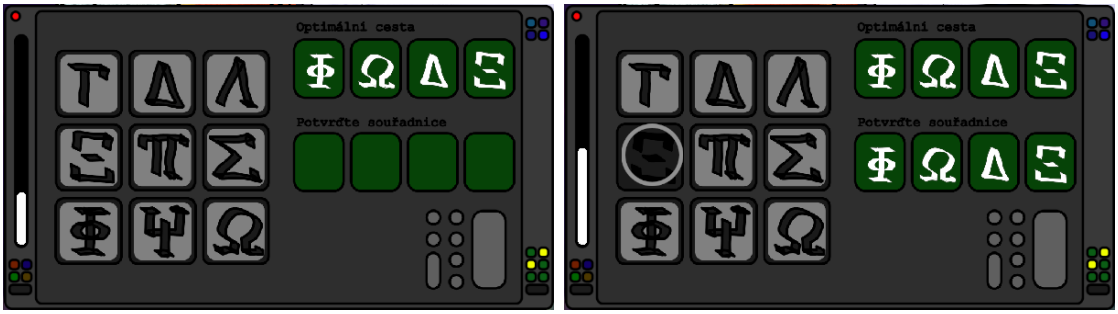
3.4 Úkoly

Úkoly jsou nejdůležitější součástí hry. Odměna za ně je velké množství skóre a penalizace je snížení integrity lodi. Každý úkol má časové omezení, ve kterém musí být splněn.

3.4.1 Souřadnice

Umělá inteligence navigačního počítače propočítala možné trajektorie a vybrala z nich tu nejvýhodnější. Loď nyní musí pokračovat na vybrané souřadnice. Ty jsou reprezentovány jako čtyři znaky z řecké abecedy. Kvůli problémům s agresivitou AI, které se dříve či později projevily po každém spuštění, není možné přenechat řízení pouze v její režii, aby nenavedla loď do nejbližšího asteroidu. A tak je před každou změnou kurzu potřeba manuální potvrzení. Aby se zajistilo, že posádka jen slepě neplní výmysly umělé inteligence, není potvrzení jen jednoduchý stisk tlačítka. Člen posádky musí souřadnice ručně přepsat.

Úkol na začátku vytvoří náhodný čtyřmístný kód. V něm se jednotlivé symboly mohou opakovat. Kód je zobrazen na displeji. Pod ním se nachází druhý, kde hráč může vidět již zadanou část kódu. Na levé straně displeje se nachází klávesnice, vytvořena jako ovládací prvek mřížka. Ta se skládá z devíti kláves, na kterých jsou některá písmena řecké abecedy.



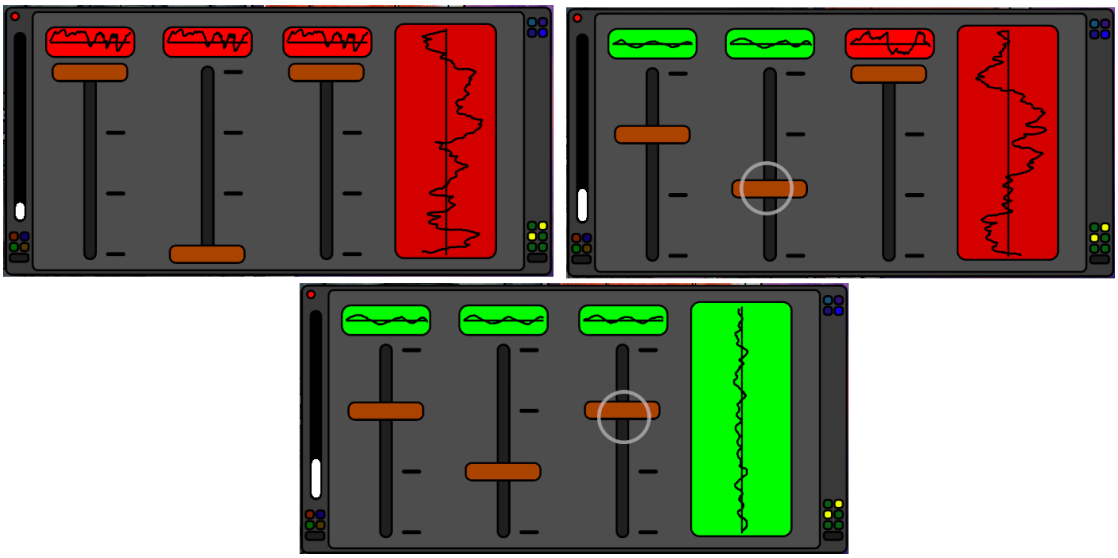
Obrázek 3.1 Úkol Souřadnice.

3.4.2 Kalibrace motoru

Motor vesmírné lodi musí z principu být velmi komplikované zařízení. Tento konkrétní model je starý a nejspíš by potřeboval vyměnit. Často se mu totiž za provozu stane, že se rozladí a tato disharmonie znatelně snižuje jeho výkon. Nastavit ho nazpátek do správného stavu není náročná záležitost, ale zato zcela nepředvídatelná. Proto je nutné manuálně vyzkoušet, které nastavení je správné. Každá ovládací páka upravuje jiný okruh a správnost jejího nastavení lze poznat z amplitud příslušných frekvencí.

Cílem tohoto úkolu je tedy nastavit všechny páky (ovládací prvek posuvník) do správné polohy. Korektní pozice páky není ukázaná a hráč se musí koukat na displej nad ní. Ten při špatném nastavení ukazuje divoce se chovající křivku s vysokými amplitudami na červeném pozadí. Hráč poté posouvá s pákou a hledá správnou pozici. Když ji najde, tak displej zezelená a křivka se uklidní. Nyní může hráč páku pustit. Tento postup opakuje pro všechny páky. Na straně úkolu se nachází displej, který ukazuje stav celého motoru. Ten obvykle ukazuje divokou

křivku, ale po správném nastavení všech pák také zezelená a křivka se uklidní. V tuto chvíli úkol končí.

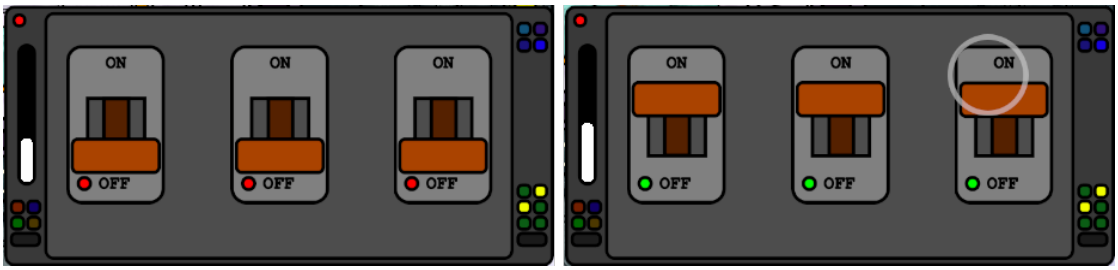


Obrázek 3.2 Úkol Kalibrace motoru. V horní řadě je možné úkol vidět v rozpracované podobě, zatímco spodní obrázek ukazuje vyřešený stav.

3.4.3 Jističe

Spadly jističe a nějaká součást lodi nefunguje. Nezávisle na tom, jestli byl přetížen kritický systém nebo někdo zapojil moc zařízení ve své kajutě, je nutné je nazpátek nahodit. Alespoň tak zní rozkazy. Co se stane potom v tuto chvíli není podstatné. Přece v nejhorším případě spadnou znova.

Před hráčem se objeví několik jističů v poloze OFF. Ty je nutné přepnout do polohy ON. Tento úkol přepracovává Jističe z minulé verze. Tam se ovládaly pouhým stisknutím, které spustilo animaci. Hráči byli často zmatení či dokonce zklamaní z toho, že je nemohli přepnout posunutím ruky. V této verzi je nutné se dotknout úchopu a rukou ho přetáhnout.



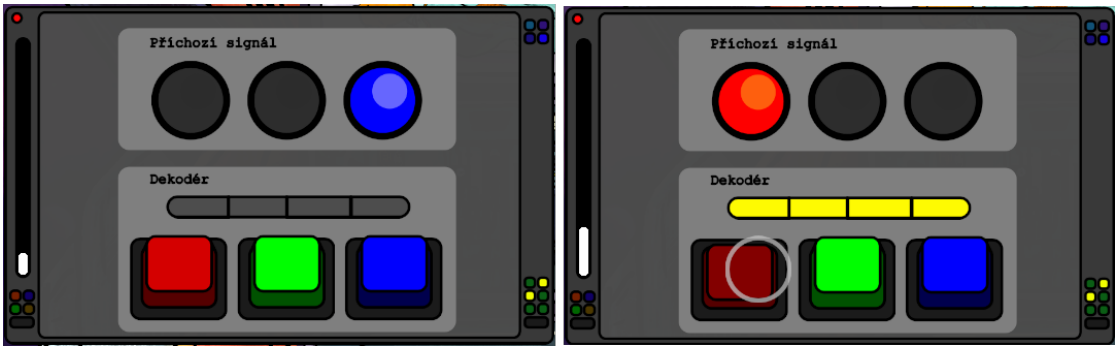
Obrázek 3.3 Úkol Jističe.

3.4.4 Dekódování

Loď je vybavena snímačem signálů z hloubi vesmíru. Podle pravidel společnosti vlastnící tuto loď je nutné prozkoumat každý příchozí signál. Dekodér se ale díky proprietárnímu rozhraní

nepodařilo propojit s přijímačem a tak je nutné kód signálu přepsat ručně. Ke zdroji signálu se naštěstí nemusí letět a tak se výstup z dekodéru pouze přepošle společnosti k dalšímu prošetření.

Jak je zmíněno v analýze v sekci 2.8.3, úkol Simon Says měl své problémy. Především špatná indikace hráči, jestli dělá něco dobře. V této verzi se princip úkolu nezmění. Cílem je stále pozorovat blikající světla a ve stejném pořadí mačkat příslušná tlačítka. Změna je v tom, že nyní se ukazuje hráči jeho postup pomocí rozsvěcujících se diod. Stisk nesprávného tlačítka všechny diody nazpátek zhasne.



Obrázek 3.4 Úkol Dekódování.

3.4.5 Zaměřování

Zdá se, že navigační počítač při hledání nejlepší cesty přehlédl jeden asteroid. Ten se nyní nachází na kolizním kurzu. Sice se z náhledu zaměřovače může zdát malý, ale to jen díky tomu, že je daleko. Umělá inteligence navigačního počítače tvrdí, že kurz je správný a před dosažením destinace odmítá přepočítat trasu. Zbývá tedy jen použít kanón a asteroid zničit.

Cílem tohoto nového úkolu je nastavit zaměřovací kříž na asteroid. K tomu slouží dva posuvníky. Jeden posouvá se zaměřovačem vlevo/vpravo a druhý zas nahoru/dolu. Poté, co hráč nastaví správné hodnoty na posuvnících, stiskne tlačítko „Vystřelit“.

Obrázek 3.5 Úkol Zaměřování.

3.4.6 Tlačítko

Každá správná futuristická vesmírná loď potřebuje spoustu tlačítek. Každé z nich zastává nějakou funkci a v jejich velkém počtu musí být problém najít to správné. V tomto úkolu se hráčům ukáže naštěstí pouze jedno. To musejí zmáčknout.

Úkol Tlačítko zůstává po vzoru Spaceship Shenanigans jednoduchý, stačí tlačítko stisknout. Ale nyní se na panelu ukazuje text, který udává funkci tlačítka. Štítek má několik variací a každá instance úkolu vybere náhodný popisek.

Obrázek 3.6 Úkol Tlačítko. V horní řadě je vidět varianta určená k házení míčkem a v řadě dolní zas varianta dotyková.

3.4.7 Lokátor

Z pozorovatelů ptáků na Zemi se při odletu do vesmíru stali pozorovatelé planet. Na mezihvězdných cestách loď prolétá skrze mnoho systémů a tito nadšenci se předhánějí, kdo najde nejzajímavější planetu. Díky vibracím motoru se ale objektiv třese a není vždy schopen zachytit planetu na první pokus.

Tento úkol je navržen pro ovládání míčkem. Hráč může trefit dvě tlačítka. Levé provede nový sken, který se ukáže na displeji uprostřed a pravé slouží jako potvrzovací. Na zobrazeném obrázku se může ukázat prázdný vesmír, nebo fotka planety. V závislosti na tom hráč buď opakuje skenování, nebo foto označí jako správné.

Obrázek 3.7 Úkol Lokátor.

3.4.8 Basketbal

Být členem posádky vesmírné lodi může být náročné. Díky regulacím je společnost, která plavidlo vlastní, nucena pracovníkům poskytnout relaxační a pohybové aktivity. Aby byly splněny kvóty dané vyhláškou, je potřeba aby, se pravidelně hrál basketbal.

Tento úkol navazuje na jeho původní verzi. Cílem je opět trefit míč do koše. Hráč hodí molitanový míček a na místě odrazu se vytvoří basketbalový míč. V nové verzi se ale koš posouvá zleva doprava, což zajišťuje větší výzvu.

Obrázek 3.8 Úkol Basketbal.

3.4.9 Společná tlačítka

Čas od času se umělá inteligence navigačního počítače pokusí připojit k systémům, ke kterým má záměrně zakázaný přístup. Tento přístup blokuje bezpečnostní protokol lodi. Avšak je možné udělit jednorázové speciální povolení, které ho přepíše. To lze provést pouze za přítomnosti dvou členů posádky. Panel pro snížení bezpečnosti obsahuje dvě tlačítka umístěná tak daleko od sebe, aby je jeden člověk nemohl stisknout současně.

Tento úkol je stejný jako ve Spaceship Shenanigans. K řešení vyžaduje míčky a cílem je trefit současně dvě tlačítka. Aby úkol nebyl příliš těžký, tlačítka zůstanou stlačená tři vteřiny po zásahu. Na rozdíl od původní verze, zde je potřeba souběžného stisku naznačena displejem, do kterého vede od každého tlačítka kabel. Ten svítí, když je tlačítko stisknuto. Z principu úkolu jsou hráči motivováni spolupracovat při řešení.

Obrázek 3.9 Úkol Společná tlačítka.

3.5 Místnosti

Při hře se může často stát, že hráč zrovna neplní úkol. Pro tyto případy je ve hře využita loď. Ta se pohupuje v pozadí a do jejích místností je vidět. V nečinném stavu mohou místnosti přehrávat jednoduchou animaci nebo pouze zobrazovat statický obrázek. V případě že se jí někdo dotkne se spustí „dotyková animace“. Po jejím skončení se místnost vrátí zpět do nečinného stavu. Takto je do hry přidáno více interaktivních prvků, které mohou upoutat pozornost kolemjdoucích. Délka

lodi se přizpůsobuje šířce obrazovky přidáváním sloupců místností. Jejich rozmístění je pokaždé jiné a náhodně vygenerované při začátku hry.

Místnosti ale mají i funkční význam. Ve Spaceship Shenanigans je čítač životů schován v horním rohu obrazovky. Je tedy mimo zorné pole hráčů a ti mu nevěnují pozornost. Umístění čítače do jiné části by mohlo problém zmenšit, ale ne eliminovat. Proto je integrita lodi (nový název pro životy) zobrazována i přímo samotnou lodí. Při každém snížení integrity se rozbijí některé místnosti. Poměr poničených místností se přibližně rovná poměru zbývající integrity. Pokud je tedy integrita na padesáti procentech, tak i půlka místností je rozbitá.

Rozbitá místnost přestane reagovat na vstup uživatele, zastaví přehrávání jakýchkoliv animací a světlo změní svou barvu na červenou. Třetí stav místnosti je zničený. Při skončení hry, když klesne integrita na nulu, se všechny místnosti postupně zničí. To je indikováno zhasnutím světla.

Kapitola 4
Softwarový návrh

4.1 Funkční požadavky

FR1 - Interaktivní stěna jako vstup Systém bude schopen přijímat a zpracovávat vstupní data z interaktivní stěny, která bude sloužit jako primární prostředek interakce uživatelů s hrou. Vstupy budou zpracovávány v reálném čase a systém bude umožňovat ovládání více uživateli najednou.

FR2 - Pravidelná tvorba úkolů V průběhu hry budou neustále vytvářeny nové úkoly. To bude zajišťovat trvalý přísun obsahu hráčům.

FR3 - Úkoly lze splnit Každý úkol bude představovat problém, který je možné nějakým způsobem vyřešit. Za splnění úkolu hráčům přibude odměna.

FR4 - Časový limit úkolu Každý úkol bude mít časový limit, ve kterém musí být splněn. Při vypršení časovače bude hráči odebrán život. Informace o zbývajícím čase bude hráči graficky znázorněna.

FR5 - Interaktivní místnosti Každá místnost uvnitř lodi bude mít vlastní animaci, která se přehraje po stisknutí. Některé místnosti mohou v nečinném stavu přehrávat jinou animaci.

FR6 - Loď reaguje na stav hry Loď, potažmo její místnosti, budou reagovat na počet životů. Při snížení jejich počtu se určité množství místností rozbije. Poměr funkčních k rozbitým místnostem bude přibližně stejný, jako poměr zbývajících životů. Při skončení hry se všechny místnosti zničí.

FR7 - Síň slávy Ve hře bude dostupný žebříček nejvyšších skóre. Do něj se hráči mohou umístit, pokud v průběhu hraní získají více skóre, než nejníže umístěný tým. Do síně slávy bude možné nahlédnout i z hlavního menu.

FR8 - Rozměry herní oblasti Velikost herní oblasti (tedy místa, kde se objevují úkoly) půjde upravovat. Herní oblast se také bude přizpůsobovat poměru stran obrazovky.

FR9 - Výběr počtu hráčů Na začátku každé hry bude možné vybrat počet hráčů, kteří se budou hraní účastnit. Systémy hry poté budou na jejich množství reagovat adekvátním způsobem.

4.2 Nefunkční požadavky

NFR1 - Unity Hra bude vyvíjena v herním enginu Unity.

NFR2 - Platforma Hra bude sestavena pro operační systém Windows.

NFR3 - Výkon Pro zajištění plynulosti bude minimální snímkovací frekvence 30 FPS.

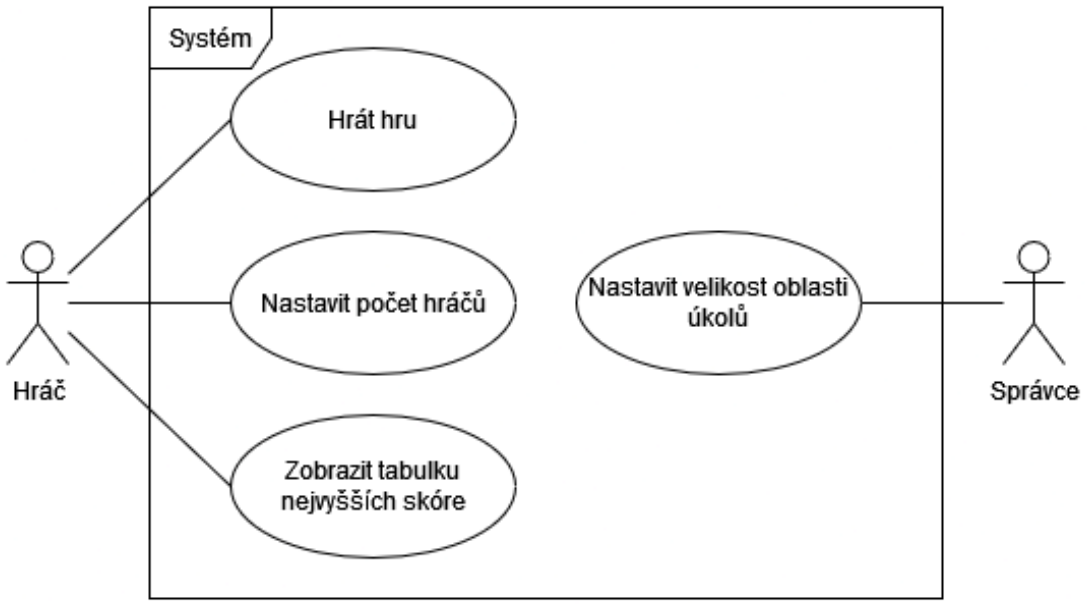
NFR4 - Přizpůsobení se velikosti stěny Celý program se bude umět přizpůsobit poměru stran obrazovky.

4.3 Aktéři

Hráč Hráčem je takový uživatel, který se účastní hraní hry a interaguje s dotykovou stěnou.

Správce Správce je uživatel, který má přístup k administrativní části hry a je zodpovědný za konfiguraci a údržbu herního systému. Správce může nastavovat parametry hry, sledovat průběh hry, a provádět údržbu systému.

4.4 Případy užití



Obrázek 4.1 Diagram případů užití

UC1 - Hrát hru Hráč přijde ke stěně, spustí hru a plní úkoly.

UC2 - Nastavit počet hráčů Hráč nastaví hodnotu počtu hráčů vhodnou pro aktuální hru.

UC3 - Zobrazit tabulku nejvyšších skóre Hráč vstoupí do síně slávy a prohlédne si tabulku nejvyšších skóre.

UC4 - Nastavit velikost oblasti úkolů Správce upraví velikost oblasti, kde se mohou vyskytovat úkoly. V tuto chvíli zle také navolit, kolik řad bude vyhrazeno pro dotykové úkoly a kolik pro házecí.

4.5 Doménový model

4.5.1 Game Manager

Tato entita zastřešuje celou hru a uchovává její stav. Ten se skládá z životů (také nazývaných jako integrita), skóre a úrovně nebezpečí. Game Manager poskytuje metody k manipulaci těchto hodnot a k jejich přístupu.

4.5.2 Task Spawner

Task spawner zajišťuje vytváření úkolů. Uchovává si mřížku, která definuje pozice úkolů, přizpůsobuje se nastavení a reaguje na poměr stran obrazovky. Má seznam všech úkolů, které může vytvořit a v různých časových intervalech se o to pokouší. Udržuje si přehled o aktuálních úkolech. Na konci hry pak všechny rozpracované úkoly zničí.

4.5.3 Task

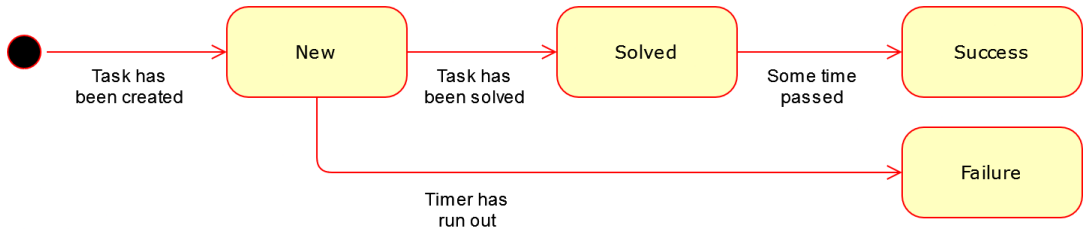
Úkol je nejdůležitější entita hry. Poskytuje hráči výzvu, kterou má překonat. Má několik vstupů a výstupů. Reaguje na změny vstupů a podle toho mění svůj vlastní stav. Ten se zas ukazuje na výstupech. Diagram přechodů stavů lze vidět na obrázku 4.2.

New Úkol byl vytvořen a hráči je umožněno ho řešit. Úvodní stav.

Solved Hráči se podařilo úkol úspěšně vyřešit. Časovač je pozastaven a hráč může vidět vyřešený úkol. V tomto stavu se úkol nachází pouze chvíli.

Success Nastává po stavu vyřešen. Hráči se ukáže množství získaného skóre.

Failure Nastává po vypršení časového limitu. Hráči se ukáže, že úkol nesplnil.



Obrázek 4.2 Diagram přechodu stavů úkolu.

4.5.4 Input

Ovládací prvek. Nabývá mnoha různých podob. Může být určen pro dotyk nebo pro míček. Vždy ale poskytuje hodnotu, kterou lze číst.

4.5.5 Output

Reaguje na změny stavů úkolu nebo ovládacího prvku. Tyto změny pak hráči graficky znázorňuje.

4.5.6 Ship Controller

Správce grafické reprezentace lodi. Při spuštění hry vytváří sestaví loď. V průběhu hry manipuluje s místnostmi v reakci na stav integrity. Na konci hry pravede zničení lodi.

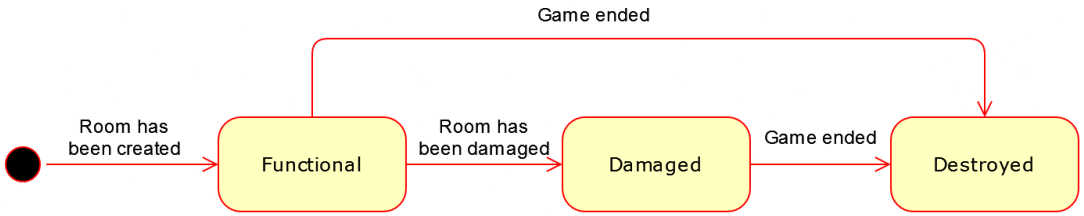
4.5.7 Room

Místnosti tvoří trup lodi. Přehrávají pasivní animaci a v reakci na dotyk spustí jinou aktivní animaci. Místnost se může nacházet ve třech stavech. Funkční, rozbitá a zničená.

Functional Úvodní stav místnosti. Místnost vypadá normálně a v reakci na dotyk přehrává animace.

Damaged Místnost se poničila při snížení integrity. Místnost nepřehrává jakékoliv animace a ani nereaguje na dotyk.

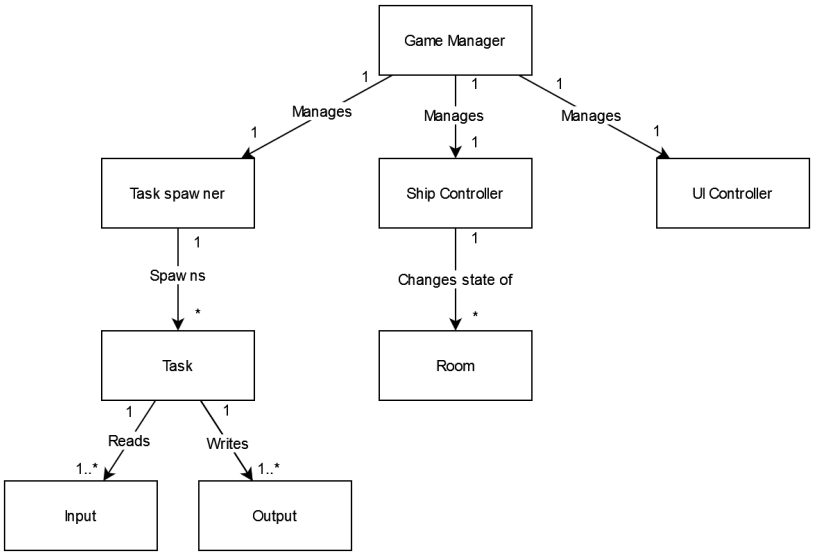
Destroyed Na konci hry se každá místnost zničí. Stejně jako ve stavu rozbitá nepřehrává žádné animace. Zničení je graficky znázorněno.



Obrázek 4.3 Diagram přechodu stavů místnosti.

4.5.8 UI Controller

Ovladač uživatelského rozhraní. Ukazuje hráči stav hry, který získá od Game Manager.



Obrázek 4.4 Diagram modelu domény.

Kapitola 5
Implementace

5.1 Nastavení projektu

Hru jsem se rozhodl implementovat v Unity ze dvou důvodů. Zaprvé mám s Unity předchozí zkušenosti a zadruhé je pro tento engine připravená knihovna, která abstrahuje komunikaci s interaktivní stěnou. Jako šablonu projektu jsem zvolil „Universal 2D“. Tato šablona obsahuje základní nastavení Universal Render Pipeline a díky tomu lze ve scénách využívat světla. Do projektu jsem importoval balíček TextMeshPro pro zobrazování textu, protože základní způsob rendrování textu v Unity je neuspokojivý.

5.2 Snímání dotyku

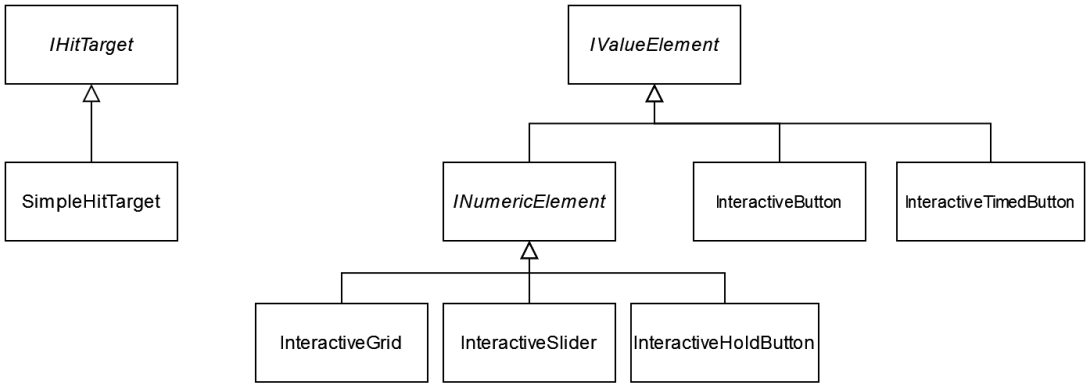
Ke snímání dotyku je využita knihovna dodaná Ondřejem Průchou ze společnosti INITI Interactive. Tato knihovna je ve formátu .dll a nelze zobrazit ani upravit zdrojové kódy. Knihovna obsahuje mnoho tříd, poskytujících různorodé funkcionality. Ke snímání dotyku je využita třída `BaseHittable`. Ta má metodu `Hit`, kterou je potřeba překrýt (override) ve třídě odvozené. `Hit` se zavolá pokaždé, když se zaznamená dotyk nad fyzikálním tvarem objektu definovaným komponenty typu `Collider2D`. Odvozené třídy pak již zajišťují konkrétní chování.

Ve hře Spaceship Shenanigans z `BaseHittable` dědí `BaseInteractive`, která rozšiřuje funkcionalitu a poskytuje další metody k překrytí. Tyto metody reprezentují dotyk, držení a uvolnění. K vytvoření ovládacího prvku tedy musí být dále děděno. Pro tento projekt jsem se rozhodl využít jiný přístup. Místo dědičnosti je využita kompozice. Vstupy využívají návrhový vzor Observer a poskytují způsob přihlášení se k a odhlášení se od událostí. Ty zůstávají stejné. Tedy událost stisknutí, událost držení a událost uvolnění. Třídy, které chtějí poskytovat vstup musí implementovat rozhraní `IHitTarget`.

`IHitTarget` pak využívají ovládací prvky. Prvky jsou schovány za rozhraním `IValueElement<T>` a přihlašují se k událostem `IHitTarget`. Zde je opět využit návrhový vzor Observer. Smyslem prvku je poskytovat hodnotu a její změny je možné odebírat. Na poslední známou hodnotu se lze i zeptat metodou `GetValue`. Ovládací prvky mohou fungovat různorodě a tak je toto rozhraní co nejvíce abstraktní. `IValueElement<T>` má specializovanější verzi `INumericElement<T>`, která navíc poskytuje informace o minimální a maximální hodnotě.

5.3 Ovládací prvky

Ve hře je implementováno pět různých ovládacích prvků. Jmenovitě Tlačítko, Časované tlačítko, Držené tlačítko, Posuvník a Mřížka. Všechny ovládací prvky mohou být libovolně posouvány, rotovány a škálovány. I přesto reagují správně. Pro zobrazení jejich stavu, například jestli je tlačítko stisknuté, je nutné využít jiných komponentů, které čtou jejich hodnotu. Typický herní objekt reprezentující interaktivní prvek má tedy nasazeny komponenty typu `IHitTarget` a `IValueElement<T>`. Jednodušší prvky mohou mít ještě vizualizační komponent a složitější prvky jsou rodiči zobrazovacích objektů. Výše popsaná třídní hierarchie je vidět na obrázku 5.1.



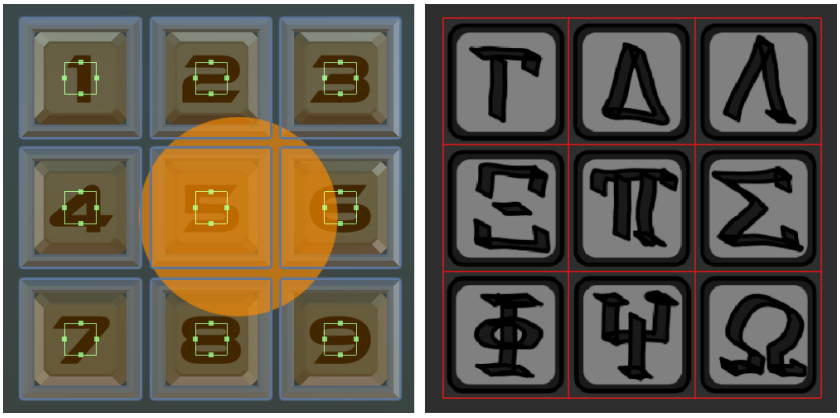
Obrázek 5.1 Diagram tříd ovládacích prvků.

V původní verzi hry jsou složitější ovládací prvky sestaveny z více jednoduchých tlačítek. To způsobuje problém u klávesnic, které nejsou dostatečně velké. Dotyk má jistý poloměr a tak se často stává, že hráč omylem stiskne více kláves najednou. Tento problém patří mezi hlavní důvody předělání Spaceship Shenanigans a přínosy aktuální verze. Na obrázku 5.2 je tento problém znázorněn. V levé části se nachází původní implementace. Každé tlačítko je samostatným ovládacím prvkem. Přestože interaktivní oblasti (vyznačena zelenými čtverci) jsou znatelně menší než vizuální reprezentace tlačítek, může jich hráč svým dotykem stisknout více najednou. Dotyk, vyznačen oranžovým kruhem, má jistý poloměr a aktivuje všechny ovládací prvky, kterých se dotýká. V pravé části je pak nové řešení. Jedna interaktivní oblast pokrývá celou klávesnici. Při dotyku se z jeho středu zjistí, do které buňky se hráč trefil. Klávesnice pak udržuje tlačítko stisknuté, dokud hráč ruku úplně neuvolní.

5.4 Úkoly

V projektu je úkol herní objekt, který má především dva komponenty. `TaskController` a `XXTask`. `TaskController` zajišťuje funkcionalitu společnou pro všechny úkoly, tedy měření času a úprava životů, potažmo skóre. `XXTask` je implementace konkrétního úkolu. Tento komponent reaguje na ovládací prvky a s využitím vzoru Observer poskytuje současný stav úkolu. Jeho prací také je při úspěšném řešení zastavit časovač.

Některé úkoly mají specifickou třídu. Například `BasketbalTask` nelze využít jinak, než pro Basketbal. Naopak úkoly, kde je potřeba zadat kód na klávesnici lze spojit dohromady. Souřadnice a Dekódování spolu sdílejí implementaci ve třídě `SequentialCodeTask`.



Obrázek 5.2 Vylepšení ovládání klávesnice. Oranžový kruh je oblast dotyku.

5.5 Správce hry

Mnoho her má nějaký hlavní skript, který řídí tok celé hry na nejvyšší úrovni. V tomto případě je to třída `GameManager`. Implementována je jako Singleton s tím rozdílem, že nezajišťuje existenci pouze jedné instance. Při výskytu více manažerů je oznámena chyba. Singleton je využit pro snadný přístup k objektu.

Tento správce zajišťuje počítání životů a skóre, zvyšování úrovně nebezpečí a začíná/ukončuje hru. Ostatní herní systémy mohou reagovat na jeho události nebo se dotazovat na konkrétní hodnoty. Například uživatelské rozhraní je v roli pozorovatele tohoto správce (viz návrhový vzor Observer, sekce 1.5).

5.6 Vytváření úkolů

`TaskSpawner` zajišťuje, aby hráči měli neustále co dělat. Před začátkem hry si vytvoří mřížku pro úkoly (hráči neviditelnou) a spočítá si jejich maximální počet. Konstanty v tomto výpočtu lze upravovat v editoru. Poté spustí smyčky, které v náhodných intervalech vytváří nové úkoly, pokud je pro ně na hrací ploše místo. Takové smyčky má `TaskSpawner` dvě. Jednu pro úkoly házecí a druhou pro dotykové. Nakonec v reakci na ztrátu všech životů zničí všechny rozpracované úkoly a zastaví jejich vytváření.

Mřížka úkolů nemusí pokrývat celou stěnu. Její velikost lze upravit změnou levého, pravého, horního a spodního odsazení. Díky tomu může být přizpůsobena konkrétním potřebám interaktivní stěny. Další proměnnou je počet řad, ve kterých se objevují dotykové úkoly. Ty jsou vždy pod házecími úkoly a ve výchozím nastavení jsou dvě. Pokud by to situace vyžadovala, je možné tuto hodnotu změnit. Jako poslední konfigurovatelná hodnota je velikost kamery. Udávaná v herních jednotkách, v základním nastavení má hodnotu pět.

5.7 Splnění funkčních a nefunkčních požadavků

Ve hře byly implementováno osm z devíti funkčních pžadavků. Funkční požadavek **FR9 - Výběr počtu hráčů** nebyl z časových důvodů doimplementován. `TaskSpawner` je ale schopen přijímat počet hráčů jako vstup a pomocí něj upravit maximální počet souběžně se vyskytujících úkolů. Aplikace splňuje všechny nefunkční požadavky.

Kapitola 6
Testování

Vzhledem k povaze tohoto softwaru byly testy prováděny manuálně, případně uživatelskými testy. Z těchto testů byla ihned získávána ústní zpětná vazba, která ovlivňovala další vývoj hry. Testování probíhalo neformálně, a proto z něj nejsou žádné záznamy. Testování zahrnovalo jak nově přidané funkcionality, tak i již existující prvky a jejich vzájemné propojení. Dále byla testována hra jako celek.

Vývoj her probíhá iterativně a během tvorby se funkcionalita a rozložení prvků často mění. Z tohoto důvodu nejsou automatické End-to-End (E2E) testy vhodné. Dalším důvodem je silné využití náhody ve hrách, což komplikuje prediktivní testování. Jednotkové testy lze využít pouze ve velmi omezených případech, protože herní prvky bývají obvykle provázané a závislé na dalších částech softwaru. Tyto vazby lze sice redukovat pomocí kvalitního softwarového návrhu a „decoupling“ návrhových vzorů, jako je například Observer. Avšak propojení nelze zcela eliminovat. Herní prvky nemohou být z principu odděleny od herního enginu. Třídy mohou například záviset na herní smyčce, kterou poskytuje herní engine, nebo na různých „singleton“ třídách, které Unity poskytuje. Příkladem mohou být třídy jako Time nebo PlayerPrefs.

Na konci vývoje bylo provedeno finální uživatelské testování. Uskutečnilo se v laboratoři ggLab, kde byla hra zprovozněna na interaktivní stěně a účastnilo se ho sedm respondentů. Účastníci dostali za úkol hru hrát a zprvu jim nebyla poskytnuta žádná pomoc. Přibližně v polovině hraní jim začalo být pomáháno s největšími problémy. Účastníci po dokončení hry dostali možnost vyplnit dotazník.

Ten obsahoval čtyři uzavřené otázky a k některým byla možnost doplnit slovní hodnocení. Otázky pokrývaly dojmy ze hry, spokojenost s ovládáním, vlastnosti hry a hodnocení úkolů. Doplňovací textové odpovědi respondenti spíše nevyplňovali.

1. **Jak hodnotíte celkový dojem ze hry?**

   První otázka v dotazníku zjišťovala pohled hráčů na hru jako celek. Odpověď byla uzavřená a hráči mohli vybírat na škále od jednoho do pěti bodů. Hráči hodnotili hru velmi pozitivně. Pět udělilo maximální hodnocení a dva zvolili 4 body.

2. **Jak se Vám líbilo ovládání interaktivní stěny?**

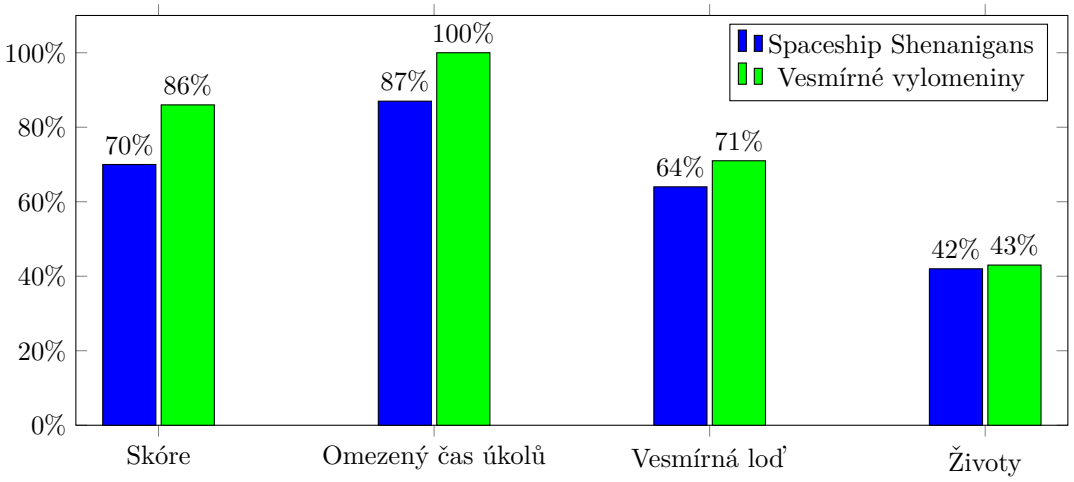
   Druhá otázka, obdobně jako první, byla uzavřená a hráči hodnotili ovládání až pěti body. Zde čtyři hráči udělili pět bodů a tři respondenti zvolili čtyři body. Otázka umožňovala přidat textovou odpověď. Toho využili čtyři respondenti. Dvě odpovědi kritizovaly kalibraci interaktivní stěny. V den testování se vyměňoval počítač na kterém se spouští aplikace pro interaktivní stěnu a správné kalibraci nebyl věnován velký důraz. Retrospektivně to byla chyba. Zbylé dvě odpovědi vytýkaly ovládání posuvníků.

3. **Zaškrtněte to, čeho jste si ve hře všiml/a.**

Tato otázka je stejná jako v dotazníku ohledně Spaceship Shenanigans (sekce 2.9). Záměrem bylo zjistit zlepšení či zhoršení komunikace hry k hráči a tím porovnat tyto dvě verze. Respondenti měli vybrat z následujících tvrzení o vlastnostech hry ty, kterých si při testování všimli.

- Na každý úkol, který se objeví, je omezený čas.
- Hra se odehrává na letící vesmírné lodi.
- Ve hře je skóre, které se na konci hry ukládá do tabulky nejlepších.
- Hra končí ve chvíli, kdy hráči nestihnout splnit pět úkolů.

Porovnání je možné vidět v grafu na obrázku 6.1. Ke zlepšení došlo u každé vlastnosti. Největší změna je u vnímání skóre, kterého si nyní všimlo 86% respondentů. Naopak u životů je zlepšení minimální a nejspíše je způsobeno rozdílným počtem respondentů, který byl v tomto dotazníku desetkrát menší.

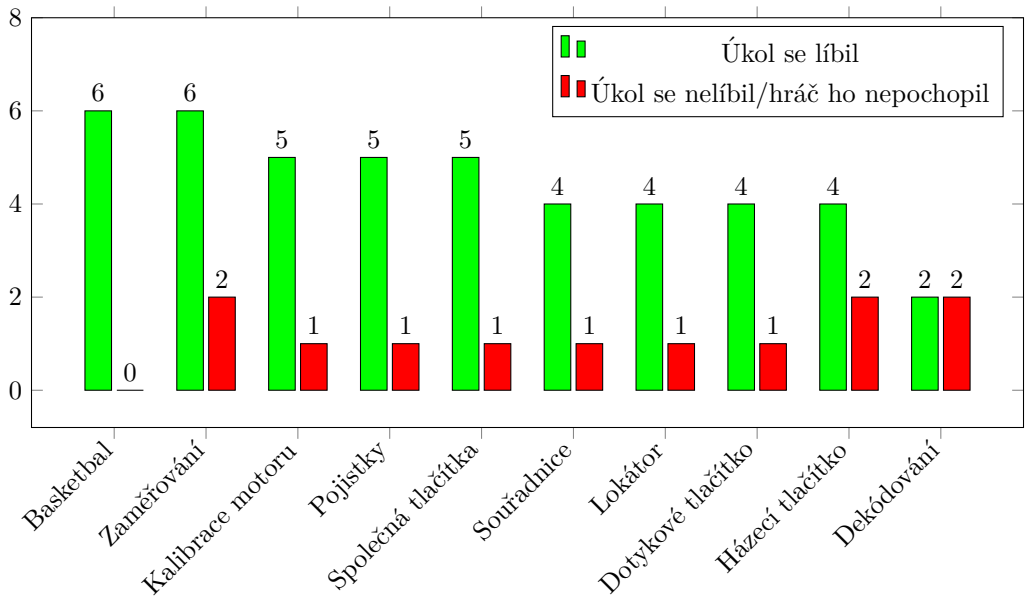


Obrázek 6.1 Graf změny vnímání vlastností hry. Hodnoty reprezentují poměr respondentů, kteří si dané vlastnosti všimli. Modré sloupce jsou z dříve uskutečněného dotazníku zpracovaném v sekci 2.9.

4. Zaškrtněte úkoly, které se vám líbily/nelíbily.

Poslední otázka byla složena ze dvou částí. Nejprve měli respondenti vybrat úkoly, které se jim líbily a poté ty, které se jim nelíbily nebo s nimi měli nějaký problém. Detailní přehled hodnot lze vidět v grafu na obrázku 6.2. Podobná otázka se nacházela i v dotazníku k Spaceship Shenanigans. Tam byly nejlépe hodnocené házecí úkoly.

V nové verzi hry zůstává úkol Basketbal mezi hráči opět nejoblíbenější. Šest z celkového počtu sedmi hráčů vyjádřilo kladné hodnocení tohoto úkolu, zatímco žádný z nich nevyjádřil negativní názor. Druhý nejoblíbenější úkol, Zaměřování, představuje zajímavý fenomén, neboť získal celkově více hodnocení, než bylo respondentů. Tato pozoruhodná situace může být vysvětlena formátem dotazníku, který respondentům umožňuje úkol označit kladně i záporně zároveň. Úkol Dekódování, který odpovídá úkolu Žárovky z původní verze hry, je opět nejméně oblíbeným úkolem. V původní verzi hry byla převažující negativní hodnocení, avšak v této nové verzi jsou negativní a pozitivní hodnocení vyrovnaná. Významné zlepšení zaznamenal úkol Kód, nyní přejmenovaný na Souřadnice. Tento úkol, který byl v minulosti spíše neoblíbený, nyní získává převahu pozitivních hodnocení díky změnám provedeným v nové verzi hry.



Obrázek 6.2 Graf oblíbenosti úkolů.

6.1 Závěr testování

Ovládání hry prošlo značným vylepšením. To bylo zjevné z pozorování hráčů při testování, kteří měli mnohem méně problémů s používáním ovládacích prvků. V původní verzi hry hráči často chtěli chytit a táhnout objekt, který tuto funkcionalitu neumožňoval. Například Jističe se přepnuly po pouhém dotyku. Nyní se jistič musí zapnout pomocí tahu ruky, což se hráčům jevilo více intuitivní.

Z dotazníku vyplývá, že hra je hodnocena kladně, ale stále má prostor pro zlepšení. Nynějším bodem úrazu jsou posuvníky. Pokud ho hráč neuchopí při prvním dotyku, posuvník přestane reagovat do úplného uvolnění. Tedy dokud hráč nezdvihne ruku několik centimetrů od stěny.

Kapitola 7
Závěr

Cílem bakalářské práce bylo vytvořit zajímavou hru, kterou lze spustit na interaktivní stěně v ggLabu. Původní verze hry byla přepracována od základu s nově nabytými zkušenostmi získaných z její tvorby a rozsáhlého uživatelského testování. Kód byl psán s úmyslem rozšiřitelnosti. Vesmírné vylomeniny jsou podstatným zlepšením od Spaceship Shenanigans především díky kvalitní implementaci ovládacích prvků, které lze využívat v mnoha různých situacích. Jejich ovládání bylo zlepšeno a problémy s nechtěnou aktivací více elementů byly odstraněny.

Vylepšení obdržely také úkoly. Ty jsou nyní více interaktivní a lépe komunikují k hráči. Například úkol žárovky neukazoval hráči, že postupuje správně. Místo toho pouze problikl, když hráč udělal chybu. Díky nekvalitnímu ovládání ale hráči mačkali špatná tlačítka nedopatřením a proto nevěděli, proč úkol bliká. Některé úkoly byly úmyslně vyřazeny (Výměna baterek), k implementaci jiných zas nezbyl čas (Jeřáb). Avšak Vesmírné vylomeniny obsahují více úkolů než Spaceship Shenanigans.

Vesmírná loď, kolem které se celá hra odehrává, obdržela také kompletní přepracování. Nyní už není jen vizuální kulisou, ale stal se z ní interaktivní objekt. Její místnosti reagují na dotyk hráče a dále rozšiřují poutavost hry.

Celá aplikace byla navržena tak, aby byla responzivní k poměru stran obrazovky. Systémy vytváření úkolů a generování lodi tyto parametry berou v potaz. Veškeré textury hry byly vytvořeny pro 4K rozlišení. Hra také obsahuje nastavení herní plochy. Díky všem těmto zlepšením je hru možné spustit i v jiných prostorech s odlišnými parametry.

Vesmírné vylomeniny mají mnoho prostoru pro zlepšení. Díky struktuře kódu je možné jednoduše přidávat nové úkoly. Mezi nápady je Požární alarm, kde hráči musejí rozbít sklo chránící spouštěč alarmu, Žhavení, kde hráč musí určitou dobu držet tlačítko a ve vhodnou chvíli ho pustit, nebo úkol Obrana, kde se hráči snaží zničit mimozemské vesmírné lodě.

Mezi další plánovaná vylepšení patří nové ovládací prvky, například ciferník nebo oblast, kde se dají přesouvat objekty. Loď může být rozšířena o další místnosti. Vhodné by také bylo doimplementování zbývajících funkčních požadavků, především výběr počtu hráčů.

Bibliografie

1. GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994. Addison-Wesley Professional Computing Series. ISBN 9780321700698. Dostupné také z: https://books.google.cz/books?id=6oHuKQe3TjQC.
2. NYSTROM, Robert. *Game Programming Patterns* [online]. 2014. [cit. 2024-05-14]. Dostupné z: http://gameprogrammingpatterns.com.
3. UNITY TECHNOLOGIES. *Order of execution for event functions* [online]. 2019. [cit. 2024-05-14]. Dostupné z: https://docs.unity3d.com/Manual/ExecutionOrder.html.
4. EPIC GAMES, INC. *AActor/Tick — Unreal Engine 5.1 Documentation* [online]. 2024. [cit. 2024-05-14]. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/API/Runtime/Engine/GameFramework/AActor/Tick?application_version=5.1.
5. GODOT COMMUNITY. *MainLoop* [online]. 2024. [cit. 2024-05-14]. Dostupné z: https://docs.godotengine.org/en/stable/classes/class_mainloop.html.
6. UNITY TECHNOLOGIES. *Scripting API: MonoBehaviour.Update()* [online]. 2024. [cit. 2024-05-14]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html.
7. GODOT COMMUNITY. *Idle and Physics Processing* [online]. 2024. [cit. 2024-05-14]. Dostupné z: https://docs.godotengine.org/en/stable/tutorials/scripting/idle_and_physics_processing.html.
8. UNITY TECHNOLOGIES. *Introduction to components* [online]. 2024. [cit. 2024-05-14]. Dostupné z: https://docs.unity3d.com/Manual/Components.html.
9. EPIC GAMES, INC. *Components In Unreal Engine — Unreal Engine 5.0 Documentation* [online]. 2024. [cit. 2024-05-14]. Dostupné z: https://dev.epicgames.com/documentation/en-us/unreal-engine/components-in-unreal-engine?application_version=5.0.
10. GODOT COMMUNITY. *Node* [online]. 2024. [cit. 2024-05-14]. Dostupné z: https://docs.godotengine.org/en/stable/classes/class_node.html.
11. LIN, Wilmer; KROGH-JACOBSEN, Thomas; ANDREASEN, Peter; BILAS, Scott. *Level up your programming with game programming patterns* [E-book]. 2022. Dostupné také z: https://unity.com/resources/level-up-your-code-with-game-programming-patterns. [Accessed 12-05-2024].
12. UNITY TECHNOLOGIES. *Object.Instantiate* [online]. 2024. [cit. 2024-05-14]. Dostupné z: https://docs.unity3d.com/ScriptReference/Object.Instantiate.html.

13. GODOT COMMUNITY. *Creating instances* [online]. 2024. [cit. 2024-05-14]. Dostupné z: https://docs.godotengine.org/en/stable/getting_started/step_by_step/instancing.html.
14. TECHNOLOGIES, Unity. *Scripting API: UnityEvent* [online]. [B.r.]. [cit. 2024-05-14]. Dostupné z: https://docs.unity3d.com/ScriptReference/Events.UnityEvent.html.
15. INITI INTERACTIVE. *INITI Playground* [online]. 2022. [cit. 2024-05-08]. Dostupné z: www.initiplayground.com.
16. BAKKER, Jason. *A GDD Template for the Indie Developer* [online]. 2009. Dostupné také z: https://www.gamedeveloper.com/design/a-gdd-template-for-the-indie-developer.
17. RIENDEAU, Danielle. *How To: Write a Game Design Document* [online]. 2023. [cit. 2024-05-14]. Dostupné z: https://www.gamedeveloper.com/design/how-to-write-a-game-design-document.

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `exe` ..... adresář se spustitelnou formou implementace
  - `VesmirneVylomeniny-build.zip` ..... sestavená hra
- `src`
  - `Scripts` ..... zdrojové kódy implementace
  - `thesis.zip` ..... zdrojová forma práce ve formátu LaTeX
  - `gitlab.txt` ..... odkaz na gitlab repozitář
- `text`
  - `thesis.pdf` ..... text práce