Středoškolská technika 2014

Setkání a prezentace prací středoškolských studentů na ČVUT

Projekt Open Game

Tomáš Dubovský, Tomáš Svoboda

Střední průmyslová škola Přerov, Havlíčkova 2

Prohlašuji, že jsme svou práci vypracovali samostatně a použili jsme pouze podklady (literaturu, projekty, SW atd.) uvedené v seznamu vloženém v práci .

Nemáme závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Přerově dne …… podpis: ……

Anotace

Projekt Open Game vznikl, aby umožnil všem programátorům, počítačovým grafikům a vlastně každému zájemci o tématiku moderního vývoje softwaru, zejména pak herního softwaru, zdarma a jednoduše prozkoumat všechny zákoutí a úskalí jednoho z nejkomplikovanějších a nejnovějších programů vůbec. Pomocí programu Unity3D otevíráme dosud velmi uzavřený a utajený svět všem bez výjimky. Díky propracovaným 3D modelům, jednoduchým a laicky popsaným skriptům, ale zároveň profesionálním nástrojům, přinášíme projekt, ve kterém si tvorbu tak náročné aplikace může vyzkoušet naprostý začátečník a přitom se nenuceně učit nárokům herního průmyslu a stejně tak i programátor profesionál, který může zrealizovat své nápady a myšlenky bez měsíců, či roků příprav. Projekt OpenGame je první svého druhu, který nijak neomezuje hranice lidské představivosti a je na každém uživateli jak tohoto projektu využije a k čemu ho použije. Možností uplatnění je stejný počet jako nápadů v lidském mozku.

Klíčová slova: OpenGame, Unity3D, programátor, laik, profesionál, neomezuje, nápad

Project OpenGame was created to allow all programmers, computer designers and indeed everyone interested in the topic of modern software development, excatly game development.

Obsah

Úvod

Na počátku 21. století se začíná neuvěřitelným tempem rozvíjet dosud téměř neznámý fenomén s názvem „počítačová hra“. Dnes se již můžeme bavit o celém průmyslu počítačových her, ať už je náš názor jakýkoliv, musíme uznat fakta, že během posledních pár let se z počítačových her staly ukazatele nejmodernějších pokroků v softwaru a se svojí komplikovaností a komplexností si co do složitosti nezadají s operačním systémem.

Dnes hry vyvíjí celé týmy programátorů, počítačových grafiků, ale i lidí jakou jsou historici, animátoři dokonce i stavební inženýři. A všichni tito lidé pracují na aplikacích, na které se rok od roku kladou stále vyšší a vyšší nároky. Je téměř neskutečné kolik peněz se vydá na takovou prostou hru a kolik lidského úsilí to stojí. Lidí, kteří by tuto problematiku rádi řešily stále přibývá, jenže naráží na nedostatek učebního materiálu, prostředků a znalostí. Všichni velcí tvůrci her si svá díla chrání a nenechají nakouknout do útrob onoho vývoje za žádnou cenu, což je celkem pochopitelné vzhledem k výrobním tajemstvím a financím investovaným do vývoje.

Proto jsme se rozhodli vytvořit projekt, který toto vše umožní, který umožní projít si a pochopit i nejzazší útroby počítačové hry, umožní jejich úpravu bez omezení a dovolí komukoliv vložit do ní své vlastní výtvory a nápady. A tak vznikl projekt OpenGame.

1 Tvorba herního prostředí

1.1 Herní engine

Na začátku každého vývoje počítačové hry se musí vytvořit program, ve kterém se budou všechny skripty, modely a vůbec práce všech dávat dohromady, který bude určovat vzhled hry a podporu různých technologií. Tedy vlastně jádro počítačové hry – herní engine. My jsme pro náš projekt vybrali herní engine Unity3D, a to ze dvou hlavních důvodů. Prvním je, že tento herní engine lze stáhnout ve verzi freeware a tak i volně šířit, což odpovídá naším požadavkům. A druhým je neuvěřitelné možnosti tohoto enginu, umožňuje přímé programování ve dvou nejpoužívanějších jazycích- C++ a JavaScript. Má spostu zabudovaných funkcí, které urychlují práci vývojáře, jako jsou CharacterController, TerrainEditor, atd. Dále má také rychlou instalaci spousty dostupných pluginů, které vytvářejí tisíce nadšenců po celém světě a tak si lze ještě více zjednodušit práci. Samozřejmě nesmíme zapomenout na okamžitou možnost importu a exportu 3D objektů, obrázků, videí a spousty dalšího, bez jakýchkoliv procedur a převodů, stačí prosté přetáhnutí do souboru a engine Unity3D už veškerou práci s komprimací a převodem odvede za vás. V neposlední řadě je to možnost kompilace hry do formátu pro všechny dnes již používané herní konzole (xbox, playstation, webový prohlížeč,mac OS a další…). Díky těmto výhodám a spoustě dalších jsme se rozhodli, že lepšího herního enginu pro tento projekt než Unity3D není.

1.2 Objekty

Engine Unity 3D obsahuje rozsáhlý zabudovaný editor objektů, ve kterém je možné velmi rychle a kvalitně vytvořit celý terén mapy bez použití jakýchkoliv jiných programů pro modelování objektů. V prostředí herního enginu je možné vymodelovat všechny základní objekty (koule, krychle, válce apod.). Na všechny modely, ať už vytvořené v herním enginu nebo vložené z jiných programů, lze vkládat textury. Dále je možné všechny modely upravovat pomocí základních modelovacích funkcí zabudovaných přímo v enginu. Všechny změny lze provádět přímo v nově vytvořené hře. Implementace fyzikálního modelu založeného na NVIDIA® PhysX®[1] je velmi snadná, stačí pouze přetáhnout model s texturami do herního enginu, který si již sám model zkomprimuje do vhodného formátu.

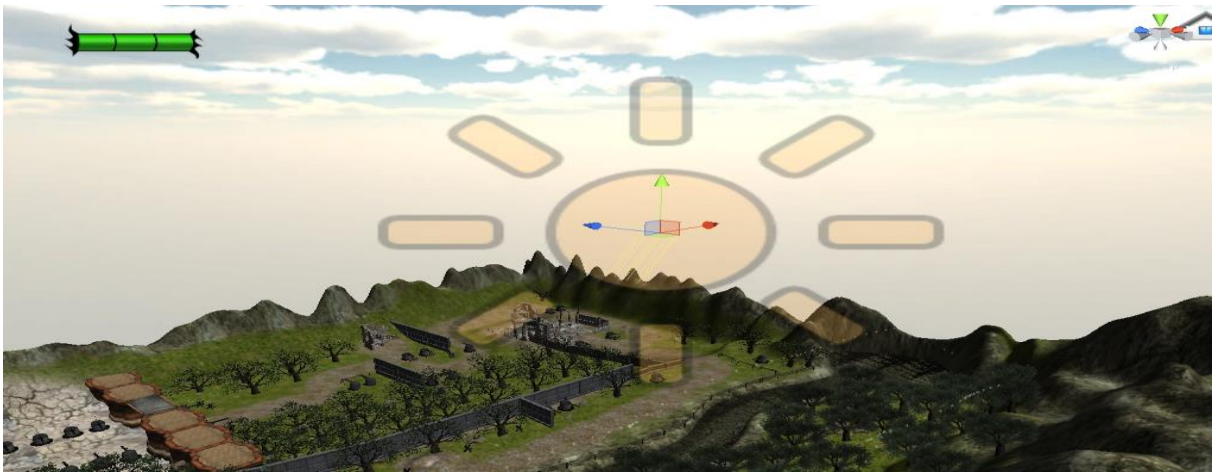
[1] NVIDIA je americká společnost specializující se na výrobu grafických procesorů a osobních počítačů. Společnost sídlí ve městě Santa Clara v Silicon Valley státu Kalifornie.

1.3 Renderování dynamického světla, implementace fyzikálního modelu

Další neméně důležitou výhodou enginu je renderování dynamického světla přímo v mapě. Renderování (anglicky rendering) je tvorba reálného obrazu na základě počítačového modelu, nejčastěji 3D. Rendering obsahuje mnoho parametrů, kterými lze ovlivnit konečný vzhled scény. Příklad světla:

1.4 Directional light

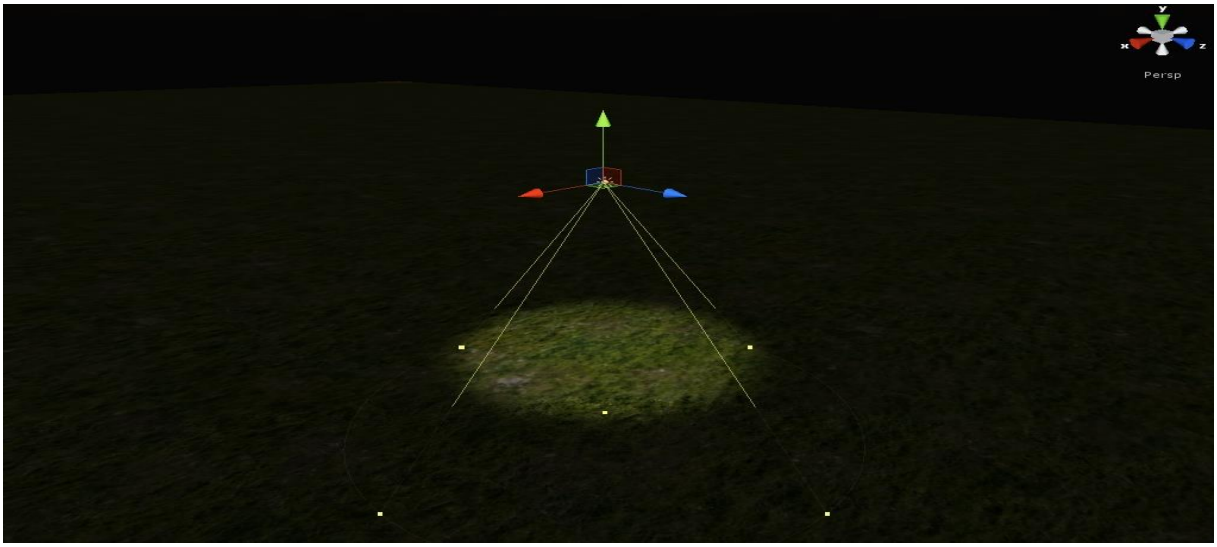
Directional light je světlo, které vkládáme do mapy, pokud chceme osvětlit celou plochu mapy. Jedná se o jaké si slunce ve hře. Stačí u něj nastavit správnou výšku, intezitu a náklon, který uzpůsobíme času ve hře.



1.5 Spot light

Spot light je světlo, které nám jak již napovídá název vytvoří světelný kužel. Toto světlo můžeme využít například u světlometů aut apod. U světla zase můžeme nastavit

intenzitu, náklon, vzdálenost ad.



1.6 Point light

Point light je doslova bodové světlo. Jedná se o světlo, které vychází z jednoho bodu do všech stran. Toto světlo nejčastěji využíváme jako osvětlení v budovách, různé druhy lamp apod. Zde zase nastavujeme intenzitu a vzdálenost, náklon je nepodstatný vzhledem k tomu, že světlo je promítáno do všech stran.

1.7 Editace animací a zvuku

Profesionální editace animací a zvuku v enginu umožňují speciální funkce přímo pro tvorbu animací pomocí keyframes[2]. Pro úpravy zvuku existuje nespočet funkcí a možností jakým stylem zvuk přehrát a upravit.

1.8 Multiplatformní engine

Slovo unity (v překladu jednota) napovídá další důležitou výhodu - schopnost enginu herní projekt jednoduše a rychle převést do formátu pro všechny dnes využívané herní přístroje. Je možné hru zprovoznit ve webovém prohlížeči, stolním počítači, na tabletu, Xboxu a playstationu. Tuto velmi širokou nabídku většina ostatních herních enginů nenabízí.

[2] Keyframes – klíčové snímky

2 Obsah projektu

Samotný projekt obsahuje aplikace Unity3D, a projekt v ní vytvořený. Tyto dvě věci jsou přílohou na DVD.

Projekt OpenGame obsahuje téměř kompletní hru v poslední fázi odlaďování. Tato hra je schválně ponechána v tomto stádiu za účelem umožnění průzkumu jejího vývoje až do posledních detailů, možnosti libovolné úpravy a třeba i kompletního vymazání a ponechání pouze určitých funkcí či modelů.

Celá hra je sestavena pomocí sady 3D modelů, které jsou volně využitelné a stejně tak volně využitelné sady skriptů. Vše je seskládáno tak, aby naprostý začátečník neztratil přehled a mohl zkoušet upravovat a učit se základním věcem. Ale zároveň byla umožněná naprostá svoboda pro profesionály, kteří si například chtějí pouze vyzkoušet určitou funkci, k čemuž je nutné mít určitou kostru hry, což by jim samotným trvalo několik let a jednalo by se pouze o přípravu zkoušky jedné jediné funkce! Z toho důvodu můžou libovolně využívat 3D modely a upravovat funkce dle vlastní představy bez omezení. Celý projekt je uzpůsoben tak, aby poskytoval co nejkomplexnější nástroj pro tvorbu počítačových her v co nejrychlejším čase a za co nejmenší náklady.

3 Programování

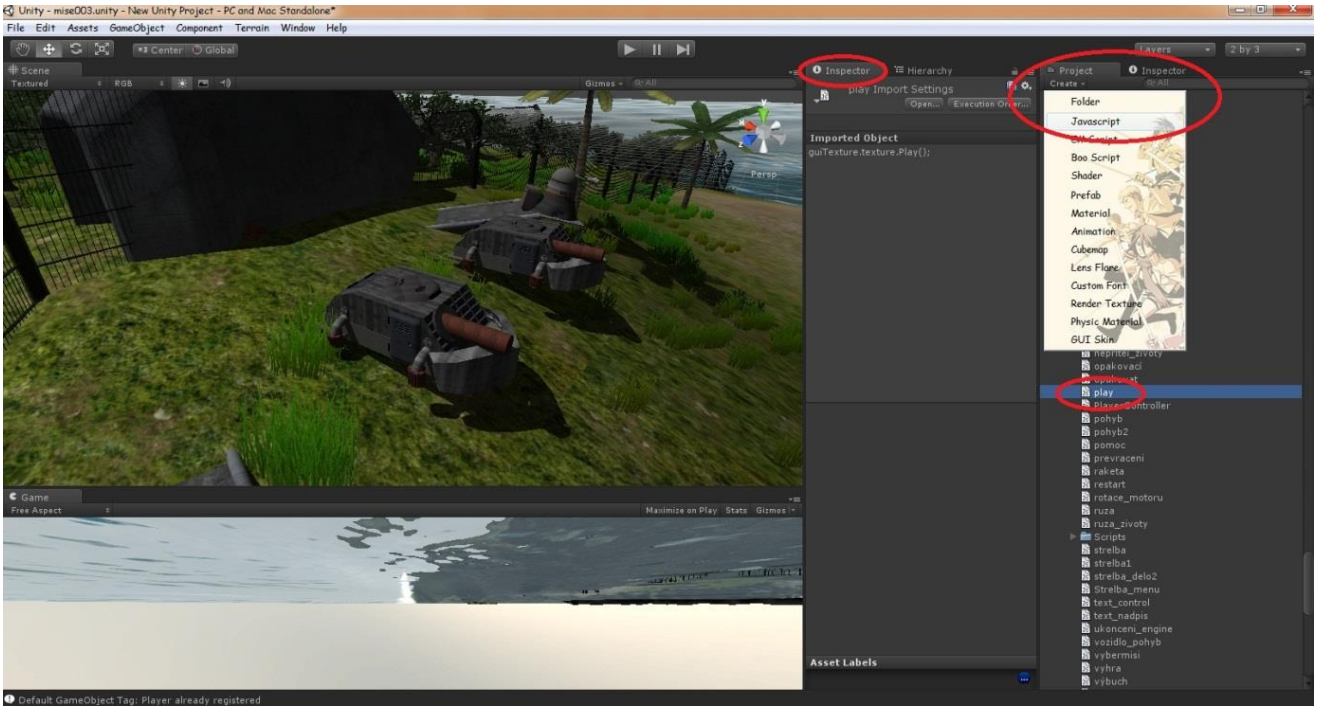
V projektu využíváme programovací jazyk JavaScript. Tento jazyk používáme kvůli našim širším zkušenostem s tímto jazykem, než-li s jazykem C++. Herní engine Unity3D podporuje oba tyto jazyky a je velmi snadné kódy s jednoho jazyka převádět na druhý. Tato možnost je tak snadná, protože většina kódu psaného v unity3D se skládá z tzv. „volání“ funkcí, které jsou součástí Unity3D.

Herní engine Unity3D nabízí přímé programování a tak nejsou nutné žádné převody mezi build_editorem a herním enginem. Hotové scripty jsou po uložení automaticky kontrolovány a tak vývojář přímo vidí každou drobnou chybu. Skripty je následně možné přímo přiřazovat k objektům.

Programování her vyžaduje znalost syntaxe funkcí pro daný herní engine. Námi vytvořené skripty představují přehled těch nejzákladnějších funkcí s ukázkou využití v praxi a s přehledným popisem. Skripty jsou naprogramovány pokud možno co nejjednodušeji a s využitím pouze základních funkcí. Je to s důvodu přehlednosti a vytvoření kvalitního základu pro možné rozšíření.

4 Vytvoření skriptu a práce v MonoDevelop

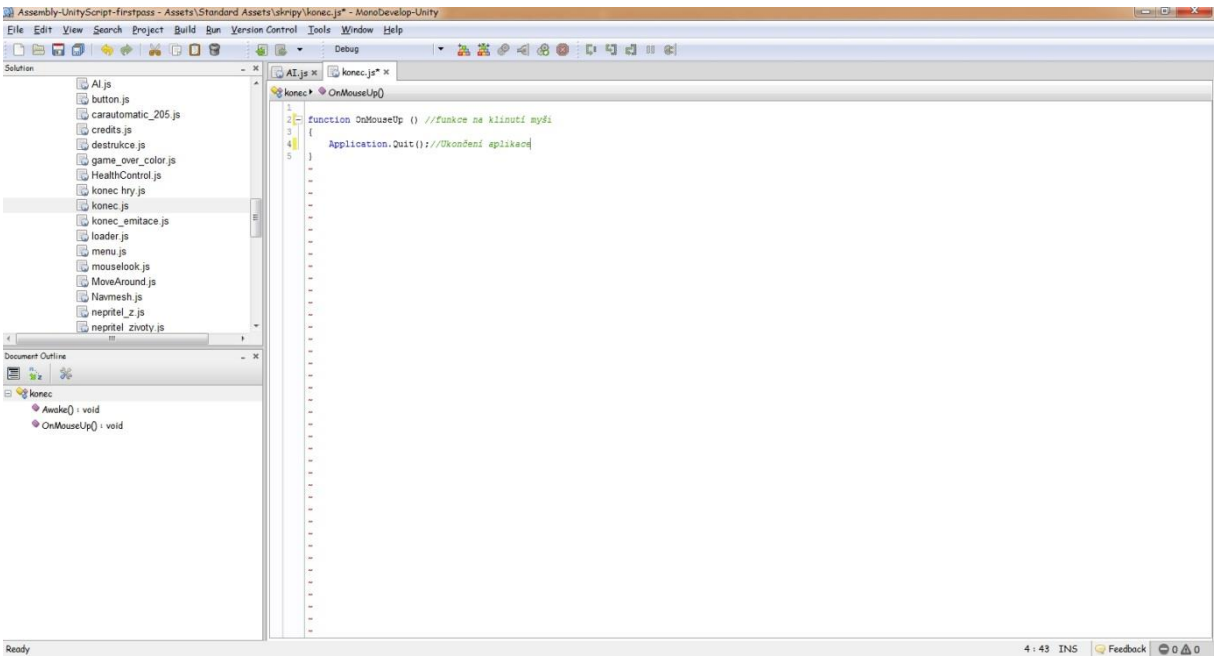
V pravém panelu „Project“, což je seznam všech věcí v našem projektu, je na horní liště tlačítko „Create“. Pokud ho rozklikneme, rozbalí se menu a zde můžeme vytvořit nový skript[3]. Ten poté můžete pojmenovat dle libosti. V panelu „inspector“ pak vidíte aktuální obsah skriptu.



Obrázek 1 Tvorba nového skriptu

Dvojitým klinutím na skript se otevře nástroj MonoDevelop-Unity, ve kterém je ho už možné přímo vytvářet, nebo upravovat. Nástroj MonoDevelop-Unity je přímo propojený s Unity3D, takže uložením scriptu se automaticky „přeloží“ a vloží do projektu. V pravém horním rohu můžete vidět seznam dostupných skriptů (viz obr. 2). Nástroj MonoDevelop-Unity poskytuje kvalitní vývojářské prostředí pro tvorbu scriptů. Ukázka prostředí MonoDevelop-Unity viz. obr. 2.

[3] Program zapsaný ve skriptovacím jazyce se označuje jako **skript**. Typicky skript těží z výhody, že se nemusí překládat, a často tvoří rozšiřitelnou (parametrickou) část nějakého softwarového projektu, která se může měnit, aniž by bylo potřeba pokaždé rekompilovat hlavní spustitelný soubor.



Obrázek 2 Prostředí MonoDevelop-Unity

4.1 Sada scriptů

V projektu OpenGame jsme vytvořily sadu základních scriptů, které by měli být návodné pro začátečníky a snadno rozšiřitelné pro pokročilé. V praxi to znamená, že jsme se snažily naprogramovat objekty pomocí základních funkcí a s využitím co nejmenšího počtu řádků. Scripty byly zároveň programovány tak, aby je bylo možné využít na další řadu her různých žánrů. Takto jsme docílily sady scriptů, kterou mohou pochopit i opravdoví začátečníci a naučit se na nich základní tvorbu scriptů v počítačových hrách.

Výhoda této jednoduchosti také spočívá v přehlednosti a právě v možnosti navázání na tento základ a rozšířitelnosti dle libosti. Sada scriptů obsahuje scripty pro pohyb, umělou inteligenci, vstupy z klávesnice a myši, ukládání a načítání, změny barev, spouštění hudby, animací, přepínání scén a spoustu dalších. Vše je tvořeno tak, aby odpovídaly nárokům

moderních počítačových her. Příklad scriptu viz. obr. 3.

```
1  var Sound: AudioClip;
2  //funkce na vstup myši
3  function OnMouseEnter ()
4  {
5      //změna barvy na...
6      renderer.material.color = Color.green;
7      audio.PlayOneShot(Sound);
8  }
9  //funkce na výstup myši
10 function OnMouseExit ()
11 {
12     //změna barvy na...
13     renderer.material.color = Color.white;
14 }
```

Obrázek 3 Script

4.2 Skripty pro ovládání hráče

Základní skriptem je ovládání hráče, tento skript zabere většinou nejvíce času a nejčastěji se celý předělává. Je to z důvodu jeho obrovské důležitosti, která spočívá v tom, že vám umožňuje vstoupit do hry a řídit danou postavu, auto atd. Náš skript je dělán tak, aby byl co nejvíce přizpůsobitelný požadavkům vývojáře a nebyl limitován pouze daným modelem hráče. Proto jsme vytvořily skript, který využívá jako vstup ovládání pomocí kláves *w,a,s,d* nebo šipek, který posunuje model určitou rychlostí za určitý časový úsek. Tuto rychlost lze měnit. Výhodou tohoto skriptu je, že nezáleží, jestli bude aplikovaný na krychli, kouli, vesmírné lodi, či závodním autě, stále bude bezproblémově fungovat. Zde je samotný skript v jazyce JavaScript:

```
//skript pro objekt vyžadující ovládání pomocí kláves

var moveSpeed : float = 5.0; //rychlost pohybu

var turnSpeed : float = 90.0; //stupně za sekundu

//poznámka: musí být použita ".0" nebo "float"!!

//Tento blok slouží pro odečtení životů, není potřebný pro pohyb!

  function OnCollisionEnter(theCollision : Collision)//funkce
zaznamená kolize colliderů
```

```
{
var other = GameObject.FindGameObjectWithTag("zivoty").GetComponent("HealthControl");
//proměnná other = najdi proměnou životy v skriptu HealthControl
}
if(theCollision.gameObject.tag == "enemyProjectile")//pokud je kolize s objektem, který má tag "enemyProjectile"
      {

    other.LIVES -=1;//odečti z LIVES 1
      }
}
//začátek funkce pohyb
function Update()
{
    // Pohyb vpřed a vzad
    // [defaultní klávesy: w/s/fwd/bak = “Vertical” (z axis)]
    var zt = moveSpeed * Input.GetAxis("Vertical") * Time.deltaTime;
    transform.Translate(0, 0, zt);

    //otáčení doleva a doprava
    //upozornění: Musí vykonávat výpočty float
    // [defaultní klávesy: a/d/left/right = “Horizontal” (x axis)]
    var turnAngle : float = turnSpeed * Input.GetAxis("Horizontal") * Time.deltaTime;
```

```
    //Input.GetAxisRaw funguje taky

    transform.Rotate (0, turnAngle, 0);

}
```

Podrobné vysvětlení skriptu je pomocí komentářů přímo v něm. Ve skritpu je implementována také funkce pro životy hráče, která funguje tak, že při zásahu vám odečte jeden život. Ve skriptu si můžete nastavit rychlost otáčení a rychlost pohybu. Veškerý pohyb je po horizontálních osách. Skript tudíž v tomto tvaru nemůže být použit pro létání, ovšem stačí zaměnit horizontální osy za vertikální, nebo přidat pohyb po vertikální ose a objekt může i létat bez delšího programování.

4.3 Systém životů

V projektu jsme se snažily vytvořit velmi jednoduchý systém životů. K tomuto účelu jsme využily funkce, která vytvoří určitý počet bloků s příkazy a mezi těmito bloky přepíná. Tato funkce je známá jako switch. Celkový počet životů jsou tři. Model hráče má ve svém skriptu na ovládání přidanou funkci, která nám zajistí, že při zásahu nepřátelskou raketou se ze skritpu kontroly životů odečte 1 a tím přepne blok, který změní stav životů. Je to velmi jednoduchý a efektivní systém, který je snadno pochopitelný a převeditelný do různých žánrů počítačových her. Zde je ukázka skriptu:

```
//Skript pro kontrolu životů

var health1 : Texture2D; //jeden život zbývá

var health2 : Texture2D; //dva životy zbývají

var health3 : Texture2D; //plný počet životů

static var LIVES = 3;//celkový počet životů

static var HITS = 0;//počet zásahů


function Update ()

{

     print("Lives:  "+LIVES+"  Hits:  "+HITS);//vytiskni  počet
životů a zásahů
```

```
      switch(LIVES)//funkce switch case přepíná GUI textury a
nakonec přepne na scénu Prohra

      {

            case 3:
                  guiTexture.texture = health3;
            break;

            case 2:
                  guiTexture.texture = health2;
            break;

            case 1:
                  guiTexture.texture = health1;
            break;

            case 0:
                  Application.LoadLevel(3);
            break;

      }

}
```

4.4 Střelba

Dalším skriptem důležitým ve většině her je skript pro střelbu. Tento skript byl sestaven tak, aby byl schopný střílet jakýkoliv model vybraný vývojářem. Je možné nastavit sílu, se kterou bude vystřelen, čas mezi jednotlivými výstřely a místo, kde bude střela vytvořena. Celý skript je udělán tak, aby se dal jednoduše upravit a přizpůsobit potřebám vývojáře. Jedná se o velmi efektivní a jednoduchý způsob střelby. Ukázka:

```
//Skript pro střelbu

var raketa : Transform;//proměnná raketa

var savedTime=3;//čas mezi jednotlivými výstřely

var force=5000;//síla výstřelu

function Update () {

var seconds : int = Time.time;//proměnná pro nastavení času

if(Input.GetButtonDown("Fire2"))//vstup myši
```

```
        {
            if(seconds!=savedTime)//pokud uplynul čas...
            {
            var raketa = Instantiate(raketa ,//vytvoř raketu v daném bodě a natoč ji podle modelu

        GameObject.Find("SpawnPoint1").transform.position,

        GameObject.Find("model").transform.rotation);
            raketa.gameObject.tag = "raketa_pritel";//přiděl raketě tag "raketa pritel"
            raketa.rigidbody.AddForce(transform.forward * force );//dej raketě sílu
            savedTime=seconds;
            }
        }
}
```

Skript je dělán tak, aby po stisknutí tlačítka myši vytvořil danou raketu v bodu tomu přiděleném a vystřelil ji přímým směrem. Ukázka hráče s přiřazeným skriptem střelba a bodem vytvoření střely viz. obr. 4.

Obrázek 4 Střelba

4.5 Skripty pro umělou inteligenci

V každé moderní hře je důležitým základem umělá inteligence. V dnešní době se touto problematikou zabývají celé týmy programátorů a vývojářů. Je to z toho důvodu, že dnešní nároky na umělou inteligenci v počítačových hrách se kladou, čím dal vyšší nároky a hráči chtějí co nejrealističtější nepřátele. V našem projektu používáme dva druhy nepřátel- statické v podobě věží a dynamické v podobě modelů. Pro oba druhy nepřátel je shodný skript AI . Tento skript je základem umělé inteligence počítačem řízených objektů v naší hře. Nejedná se příliš složitý skript, takže i inteligence nepřátel je tomu odpovídající, ovšem pro naše účely to bohatě stačí. Můžeme v něm nastavit cíl, který bude nepřítel sledovat, objekt, který bude střílet, čas mezi výstřely, dosah aktivity nepřítele. Ukázka skriptu:

```
//AI skript

var LookAtTarget:Transform; //proměnná "dívej se na cíl" vyber cíl, který bude objekt následovat

var damp = 6.0;//proměnná utlumení (slouží pro utlumení rychlosti otáčení, je to s důvodu větší reálnosti otáčení

var bullitPrefab:Transform;//proměnná střela

var savedTime=0;// proměnná ušetření času

var range = 2;// proměnná dosah (slouží k aktivaci skriptu až po dosažení dané vzdálenosti)

var myTransform : Transform;// proměnná tento objekt( slouží k určení aktuální pozice objektu)


function Update ()
{
     var distance = (LookAtTarget.position - myTransform.position).magnitude;//určuje danou vzdálenost od cíle(rozdíl mezi pozicí cíle a objektu)

     if (distance < range) {//pokud je vzádelonost od cíle menší jak určená vzdálenost pro zahájení akce

     if(LookAtTarget)// pokud sleduje cíl
```

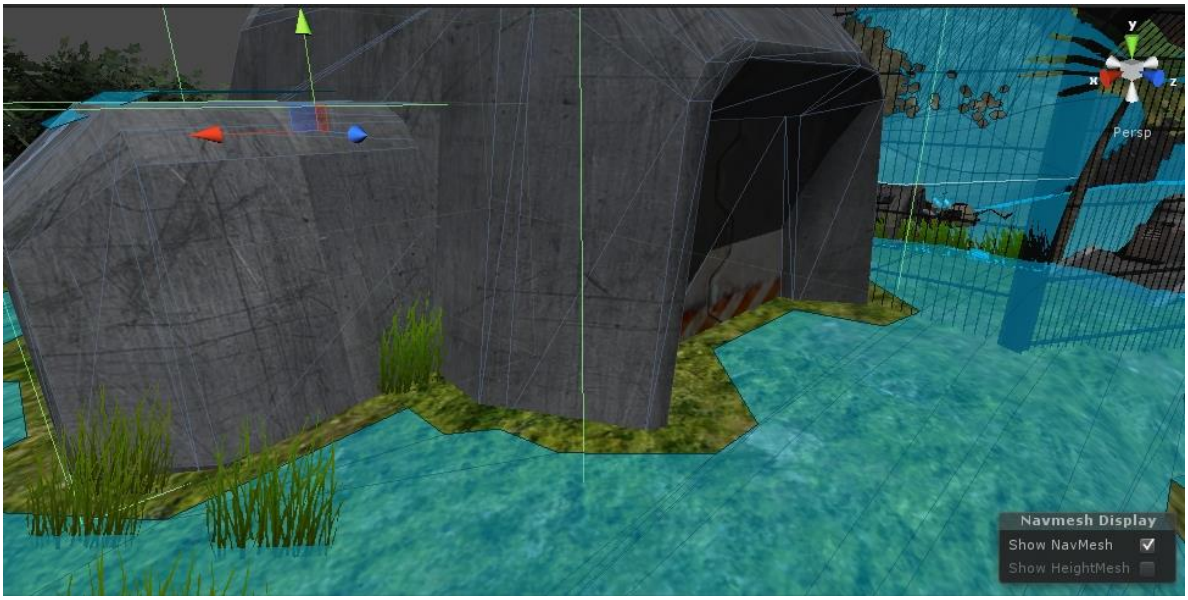
```
	{
		var rotate = Quaternion.LookRotation(LookAtTarget.position - transform.position);// určuje rotaci čelem na cíl s daným spomalením

		transform.rotation = Quaternion.Slerp(transform.rotation, rotate, Time.deltaTime * damp);
		var seconds : int = Time.time;//určuje čas v sukundách, využívá funkce Time

		var oddeven = (seconds % 2);

		if(oddeven)

		{
		Shoot(seconds);

		}
	}
}
}

function Shoot(seconds)//funkce střelba
{
	if(seconds!=savedTime)//pokud sekundy=ušetřený čas
	{
		var bullit = Instantiate(bullitPrefab ,transform.Find("spawnpoint").transform.position ,transform.Find("spawnpoint").transform.rotation);//vytvoř objekt střelby
		bullit.gameObject.tag = "enemyProjectile";//dej mu "tag" enemyProjectile

		bullit.rigidbody.AddForce(transform.forward * 1000);//dej střele "sílu"

		savedTime=seconds;
	}
}
```

Objekt s tímto skriptem začne reagovat, pokud se hráč dostane do určené vzdálenosti od něho. V ten samý okamžik se na hráče tento objekt otočí a zamíří na něj. Následuje střelba danými projektily s daným časovým rozestupem. Pokud hráč opustí tento prostor, střelba ustane. Ukázka statického nepřítele viz obr.5.

Obrázek 5 Statický nepřítel

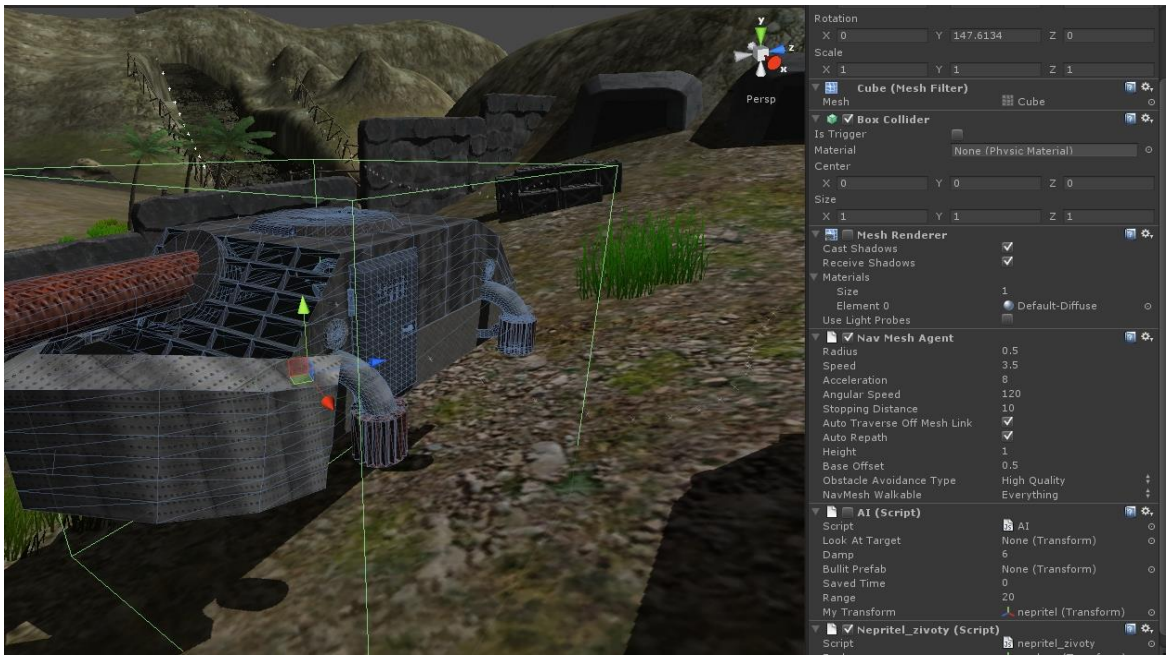
Toto ovšem stačí pouze u statického nepřítele, který je připoután k danému bodu a nepohybuje se. Pokud chceme využívat dynamických nepřátel, je potřeba přidat skripty, které nám zajistí pohyb a navádění nepřítele za hráčem. Tato umělá inteligence už je mnohem složitější, ovšem výsledkem jsou pohybující se nepřátelé.

Herní engine Unity3D nám tuto práci usnadňuje funkcí NavMesh. Tato funkce nám přes celou mapu vytvoří mapu navigace, což je mapa, po které se může nepřítel pohybovat. Tu si můžeme přizpůsobit dle svých požadavků. Například pokud nechceme, aby se nám nepřítel zasekl o nějakou zídku a musel ji objet, nastavíme ji jako statický objekt a tím se v mapě navigace plocha zídky odstraní, takže nepřítel ji bude muset objet. Pro názornost screenshot mapy navigace viz. obr. 6.



Obrázek 6 Mapa navigace

Další funkcí, která se k navigaci váže je NavMeshAgent. Tato funkce od vývojářů Unity3D zajistí navigaci samotného objektu. Je zde mnoho nastavení, mezi ty základní patří rychlost, zrychlení a vzdálenost zastavení. Tuto funkci používáme, protože se vývojářům opravdu povedla a je velmi jednoduchá na pochopení i pro začátečníky. Ukázka dynamického nepřítele s funkcí NavMeshAgent viz. obr. 7.



Obrázek 7 Dynamický nepřítele s funkcí NavMeshAgent

Ukázka skriptu NavMesh:

```
//NavMesh skript

var target : Transform;//cíl
var LookAtTarget:Transform;//dívej se na cíl
var range = 2;//dosah
var myTransform : Transform;//proměnná tento objekt( slouží k
určení aktuální pozice objektu)
function Update () {

      var      distance      =      (LookAtTarget.position      -
myTransform.position).magnitude;//určuje danou vzdálenost od
cíle(rozdíl mezi pozicí cíle a objektu)

      if (distance < range) {//pokud je vzádelonost od cíle menší
jak určená vzdálenost pro zahájení akce

      if(LookAtTarget)
            {
                  GetComponent(NavMeshAgent).destination          =
target.position;//přidej funkci NavMeshAgent
            }
      }
}
```

Skritp nám zajistí, že nepřítel sleduje námi zadaný objekt, a to dokud objekt nedosáhne hranice pro spuštění skriptu. Poté spustí funkci NavMeshAgent.

4.6 Životy nepřátel

Poslední věcí, kterou bylo třeba vyřešit u nepřátel, jsou jejich životy. U statických nepřátel je v projektu použito jednoduchého skriptu na pouhou destrukci po jednom zásahu. Tento skript využívá animace exploze, což je animace ohně, který na sobě však má skript, aby hořel pouze určitou dobu. Takže po zásahu se objeví plameny, které nějakou dobu hoří a následně zmizí. Spolu s objevením plamenů zmizí i objekt nepřítele. Ukázka skriptu pro zmizení plamenů:

```
//konec emitace skript

var lifeTime = 30.0; //čas života

var startTime = 0.0; //startovací čas

function Awake(){
```

```
   startTime = Time.time;

   Invoke ("StopMe", particleEmitter.minEnergy * .5F);//vyvolej
stopMe když je energie částic vynakládána

   }

function StopMe () {//funkce StopMe

  particleEmitter.emit = false;// Zastav animaci

}
```

Tento skript po uplynutí 30s vypne animaci.

Ukázka skriptu destrukce:

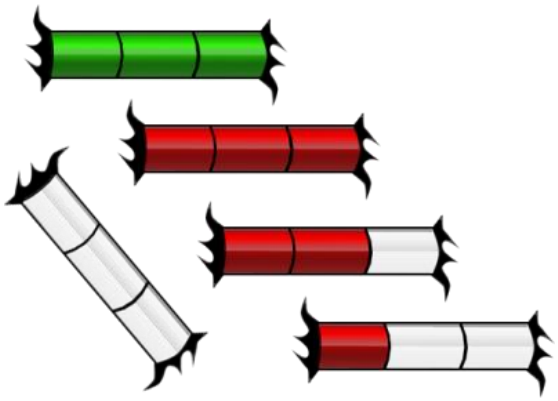
```
var exploze:Transform;//proměnná exploze

function OnCollisionEnter(theCollision : Collision) {//pokud je
zaznamenána kolize

     if(theCollision.gameObject.tag == "raketa_pritel")//pokud se
jedná o kolizi s objektem, který má tag "raketa pritel"

     {

          Instantiate(exploze,transform.position,
transform.rotation);// vyvolej explozi

          Destroy(gameObject);//znič objekt

     }

}
```

Pokud je zaznamenána kolize objektu s raketou hráče zničí daný objekt.

4.7 Výhra a prohra

Jako v každé hře je potřeba zajistit možnost výhry a stejně tak i prohry. V projektu Open game využíváme skript, který počítá dané objekty a pokud nenalezne žádný ukončí hru a přepne ji do scény „výhra“. Je to velmi jednoduché a nenáročné řešení, ale zároveň velmi efektivní a hráčsky zajímavé. Prohra je řešená třemi životy hráče. Tyto životy jsou graficky znázorněny GUI texturami. Skriptem prohry je výše uvedený skript kontroly

životů. Celý systém je velmi jednoduchý ovšem chytlavý. Tento systém nezatěžuje systém a je snadno pochopitelný. Ukázka GUI textury životů:



Využití GUI textur je s důvodu jejich praktičnosti, pomocí nich lze názorně zobrazit aktuální status životů. Zobrazení je jednoduché a přitom velmi přesné. Ukázka skriptu pro výhru:

```
//skript výhra

var targetTag = "enemy";//proměnná objekt s tagem "enemy"

function Update () {

    if(GameObject.FindWithTag(targetTag) == null){//pokud nenajde žádný objekt s tagem "enemy"

        print("none left");//tisk

        Application.LoadLevel (4);//načti scénu


    }

    else{//nebo

        print("enemies left");//tisk

    }

}
```

Skript jednoduše funguje na neustálé kontrole počtu nepřátel a při počtu nula ukončí hru.

Po ukončení mise následuje načtení nové scény, pro výhru i prohru jsou to odlišné scény. Každá scéna je výjimečná tím, že používá pokud možno co nejvíce 3D prostorových objektů, hra je tak mnohem interaktivnější a působí lepším dojmem. Zároveň ukazujeme, že scény jdou vytvořit poměrně velmi jednoduše a díky našim objektům není ani pro naprostého začátečníka problém vytvořit scénu, která vypadá jako z profesionálního studia. K vytvoření takové scény je zapotřebí pouze několik 3D textů sloužících k navigaci, nějaké modely a pozadí, popřípadě hudba na pozadí. Scény v projektu OpenGame může následně kdokoliv upravit a vyměnit pouze texturu v pozadí a vzniká scéna nová. Ukázka scény prohra:

Ve scéně je jeden model zničeného hlavního modelu, v pozadí textura zničeného města a celkem tři 3D texty. Dva z nich jsou aktivní pomocí skriptů a po najetí na ně myší změní barvou, po kliknutí vás přesměrují na danou scénu. Celá scéna je ještě doplněna hudbou a vytváří tak efekt tragédie. Je to jeden ze způsobů jak vytvořit zajímavé a netradiční scény. Scéna výhry je řešena podobným způsobem.

4.8 Doplňující skripty:

V každé hře je nespočet skriptů, které zajišťují na první pohled triviální a mnohdy téměř zbytečné funkce. Ovšem pravdou je, že právě tyto skripty vytváří ten pravý dojem a dodávají hře ten pravý efekt. Mezi tyto skripty patří například skripty na změnu barvy textu, spouštění videí, pozastavení scény, restart apod…Díky těmto skriptům stoupá vizuální stránka hry a náš projekt umožňuje začátečníkům i pokročilým nakouknout i do těchto zákoutí

moderních počítačových her a nabídnout inspiraci pro vlastní tvorbu. Skript pro změnu barvy textu:

```
//změna barvy textu

var Sound: AudioClip;

//funkce na vstup myši

function OnMouseEnter ()

{

      //změna barvy na...

      renderer.material.color = Color.green;

      audio.PlayOneShot(Sound);

}

//funkce na výstup myši

function OnMouseExit ()

{

      //změna barvy na...

      renderer.material.color = Color.white;

}
```

Skript funguje na vstup myši, přičemž mění barvu při vstupu kurzoru na text a mění barvu zpět při výstupu.

4.9 Scény

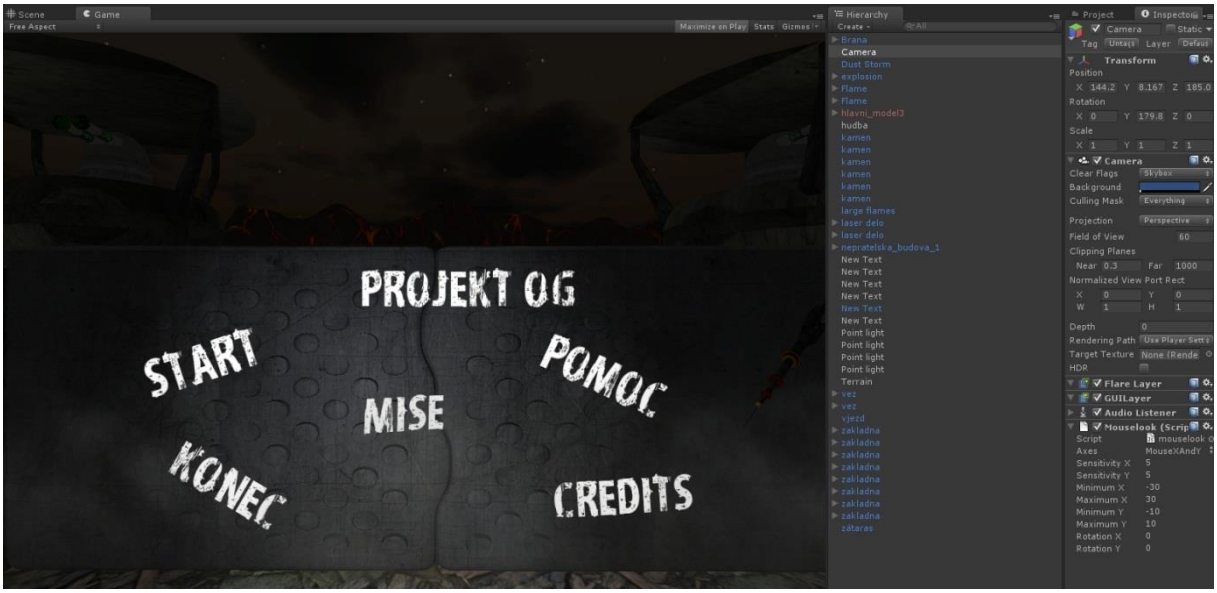
Scéna je nové prostředí, ve kterém programátor pracuje. Ze scén se pak skládá celý projekt. Každá scéna je ve své podstatě samostatné pracovní prostředí, scény mohou být naprosto rozlišné. Scény jsou navzájem propojeny pomocí skriptů. Je to moderní a velmi efektivní řešení. Na každou scénu může vývojář použít různé modely, skripty, animace...atd. a nemusí mít strach, že by jakkoliv ovlivnil ostatní scény.

4.10 Hlavní menu

Hlavní menu je jedna z nejpodstatnějších částí počítačové hry. Hráč tuto scénu uvidí jako první a bude se do ní neustále vracet. V našem projektu využíváme nejnovějších trendů interaktivních 3D menu. Tyto menu využívají 3D objekty a různé animace namísto pouhých textur a tak vtahují hráče ještě více do hry. Na těchto menu je zajímavé prolínání vizuálních efektů a snadné funkčnosti. Ukázka scény hlavního menu:

Scéna samotná se snaží vystihnout ráz hry a zaujmout hráče. K tomuto účelu jsou do scény stylizovány hlavní modely ve hře. Prostředí mise podtrhují animace vířícího prachu a pohybujícího se světla. Vývojář začátečník se může na této scéně naučit základní strategii při sestavování menu, objevit kouzlo drobných animací a ovládání textu. Novinkou v tomto menu je možnost pohybu kamery, což ještě podtrhuje interaktivitu. Tento pohyb je realizován pomocí funkce MouseLook. V této funkci lze nastavit rozsah pohybu kamery a to po všech osách a samozřejmě citlivost pro vstup z myši. Ukázka hiearchie objektů ve scéně a nastavení

funkce MouseLook:



4.11 Výběr misí

Pro výběr misí slouží scéna k tomu určená, v této scéně máte možnost vybrat si jednu ze tří hotových misí. Scéna je udělána v podobném stylu jako hlavní menu. Každý 3D text má za sebou objekty a prostředí podobné dané misi. Hlavním rysem této scény je její přehlednost a účelnost. Žádný s hráčů nechce strávit den hledáním své oblíbené mise. V naší scéně má vše uspořádané a přehledné. Tato scéna slouží jako perfektní inspirace pro vývojáře v jejich vlastní tvorbě. Pomocí malého počtu objektů tak mohou vytvořit zajímavé herní prostředí. Ukázka:

5 Herní mise:

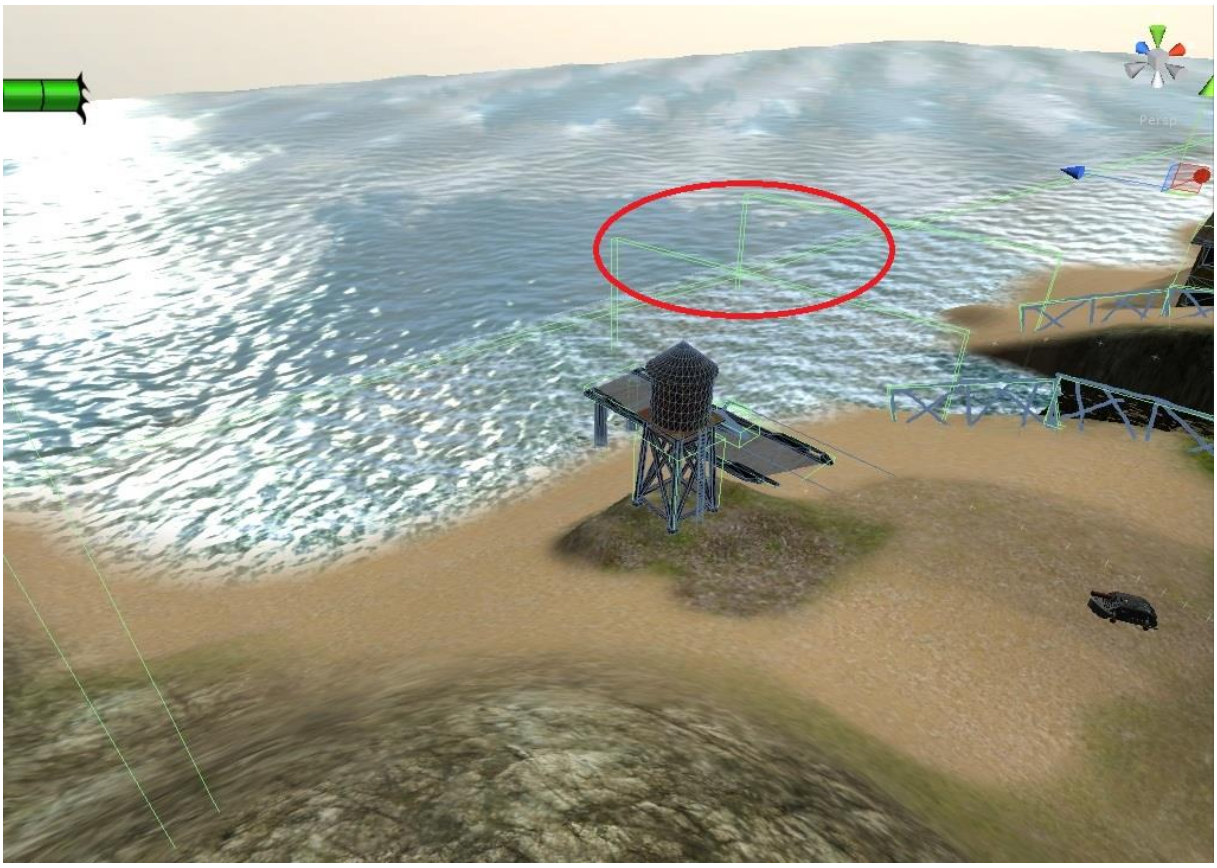
Herní mise jsou tou nejpodstatnější částí hry, protože v nich hráč stráví nejvíce času a v těchto misích se teprve ukáže funkčnost skriptů a design modelů. Náš projekt má takové mise tři. Každá z nich má svůj vlastní design a sadu modelů. Vývojář si je může upravit a předělat podle svého, ale základ má již hotový. Dynamičnost, různorodost herního světa, interaktivita, plynulost…to jsou dnes hlavní měřítka pro hodnocení hry. Náš projekt vývojářům ukazuje jak toho dosáhnout a jak vytvořit hru s výborným hodnocením. Různorodost misí nabízí programátorům možnost vyzkoušet si své skripty na široké řadě modelů. Naopak široká sada skriptů pro každou misi nabízí tvůrcům 3D modelů možnost vyzkoušet své modely v mnoha situacích a herních prostředích. Pro pouhého zájemce bez zkušeností nabízí mise možnost provedení nesčetných změn v nastavení skriptů či modelů a následně sledovat jak se dané změny projeví na hře a tím se nenuceně učit. Ukázka jednotlivých misí podle pořadí první, druhá, třetí:


Persp


Persp

5.1 Neviditelné collidery

Celá mapa je ošetřena takzvanými neviditelnými collidery. Jedná se o collidery, které jsou rozmístěny na místech, kde by mohl případně hráč vyjet z mapy a zaručují, že se bude držet na dané trase. Samozřejmě jedná se pouze o několik colliderů, protože naražení do neviditelné stěny není moc realistické. Proto je v mapě spíše využito objektů typu plot, zábrana apod., které splňují stejný účel, nicméně vypadá to mnohem lépe, když narazíte do plotu, než do neviditelné zdi.



Závěr

Projekt OpenGame je mnohem více než pouhá hra. Je to nástroj, který umožní všem zájemcům prozkoumat všechny taje moderních her. Slouží jako nástroj, rádce a pomocník při realizaci vlastních projektů. Svou otevřeností a jednoduchostí může zaujmout i úplné začátečníky. Projekt OpenGame se bude nadále rozvíjet, jelikož jeho hranice nejsou nijak omezeny.

Použité zdroje

1. Unity Technologies. *Unity 3D* [online]. [cit. 2014-18-03]. Dostupné z: http://unity3d.com/unity/Unity 3D

2. Autodesk: Produkty řady Autodesk 3ds Max. *Autodesk, Inc.* [online]. [cit. 2014-18-03].

3. Windows: Windows Live Movie Maker. *Windows* [online]. [cit. 2014-18-03]. Dostupné z: http://windows.microsoft.com/cs-CZ/windows7/products/features/movie-maker

4. Softonic.com: Total Video Converter 3.71. *Softonic.com* [online]. [cit. 2014-18-03]. Dostupné z: http://total-video-converter.en.softonic.com/

5. Six Times Nothing: River Tool. *Six Times Nothing.com* [online]. [cit. 2014-18-03]. Dostupné z: http://sixtimesnothing.com/river-tool/

Seznam obrázků

Seznam příloh

DVD