ODBORNÁ MATURITNÍ PRÁCE

Obor č. 18: Informatika

Realizace 3D hry z pohledu z první osoby s RPG prvky

Autor: Tomáš Krejčík

Škola: SPŠE Ječná, Ječná 30, 121 36, Praha 2

Kraj: Hlavní město Praha

Konzultant: Mgr. Alena Reichlová

Praha 2022

Prohlášení

Prohlašuji, že jsem odbornou maturitní práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.
Nemám 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 Praze dne 25. 3. 2022 ……

Poděkování

Rád bych tímto poděkovali Mgr. Aleně Reichlové za vedení mé dlouhodobé maturitní práce a panu Bc. Janu Pavlátovi za oponenturu.

Obsah

Anotace

Tento dokument popisuje postup a detaily vývoje mé dlouhodobé práce, včetně vytváření vlastních modelů a scriptech funkcionalit v jazyce C#. Mým cílem bylo vytvořit 3D hru z pohledu první osoby s funkcemi z role – playing her.

Úvod

Zadání

Jako téma mé dlouhodobé práce jsem si vybral 3D hru z pohledu první osoby s RPG prvky. Hráč bude mít v arzenálu pár zbraní, se kterými bude moct porážet různé nepřátele. Hráč bude ničením nepřátel získávat zkušenosti a měnu, za které si pak následně může vyzkoumat a koupit různé vylepšení i nové schopnosti. Všechny modely budou ručně dělané v 3D modelovacím softwaru.

Inspirace

Již při vybírání tématu velice přispěla má ročníková práce ve třetím ročníku, která byla také vytvořena v enginu Unity3D. Důvodem byly již vytvořené systémy a funkce, které mi ušetřili čas na vytváření této práce.

Název

Každý produkt by měl mít nějaké jméno, které ho bude reprezentovat, ať už se jedná o obraz, písničku, film nebo v tomto případě hru. Když přišlo na vymýšlení názvu, tak mi to netrvalo déle než pár minut. Společně se spolužákem jsem vymysleli každý jedno slovo, které jsem následně spojily a vznikl z toho Meta Mayhem. Poté jsem našel font FROSTBITE který jsem použil pro logo a následně i na několika místech v samotné hře.

Software

Unity

Herní engine, který je základem naší hry. Používal jsem jeho zdarma edici. Začal jsem s verzí 2021.2.0f1, ale uprostřed vývoje jsem přešel na novější 2021.2.13f1. Ve hře jsem využil i balíčky přímo od firmy Unity: Input System, Animation Rigging, 2D PSD Importer, Universal Rendering Pipeline a Visual Effect Graph.

Blender 3.0

Pro modely jsem použil open source program blender.

Visual Studio 2022

IDE ve kterém jsem psal všechny scripty. Používal jsem jeho Community verzi, která je zdarma.

Gimp

Zdarma open source program na úpravu fotek. Využil jsem ho pro vytvoření několika základních textur.

Audacity

Open source program ve kterém jsem nahrával a následně upravil všechny použité audio soubory ve hře.

GitHub Desktop

Počítačová aplikace GitHubu, používající verzovací nástroj Git, kde jsem ukládal a sdílel můj projekt.

BFXR

Jednoduchý open source program pro vytváření „8-bit“ zvukových efektů.

Hra

Modely

V plánovací fázi jsem přemýšlel nad v jakém stylu chceme naši hru vytvářet. Nakonec jsem se rozhodl pro futuristické zasazení.

Zbraně

Těžká zbraň

Inspirována brokovnicí ze série DOOM.

Pistole

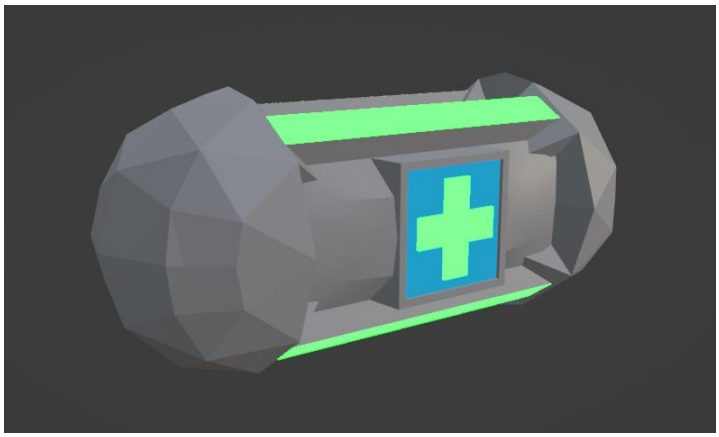
Vymodelovaná podle reálné zbraně Sig Sauer P320 a její futuristicky vylepšená verze.

Kulomet

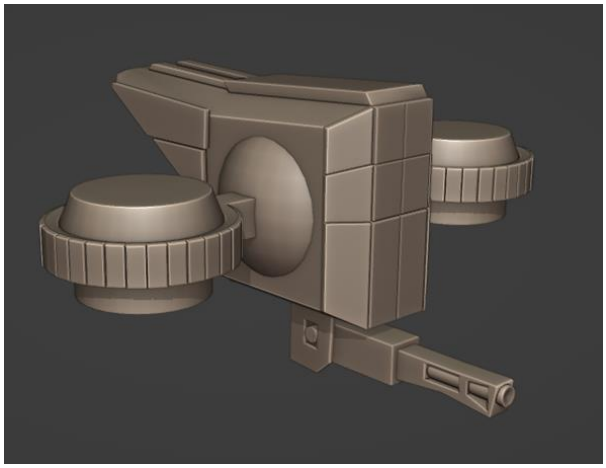
Inspirovaný reálnou zbraní FN Minimi/M249 a jeho futuristicky upravená verze.

Ostatní

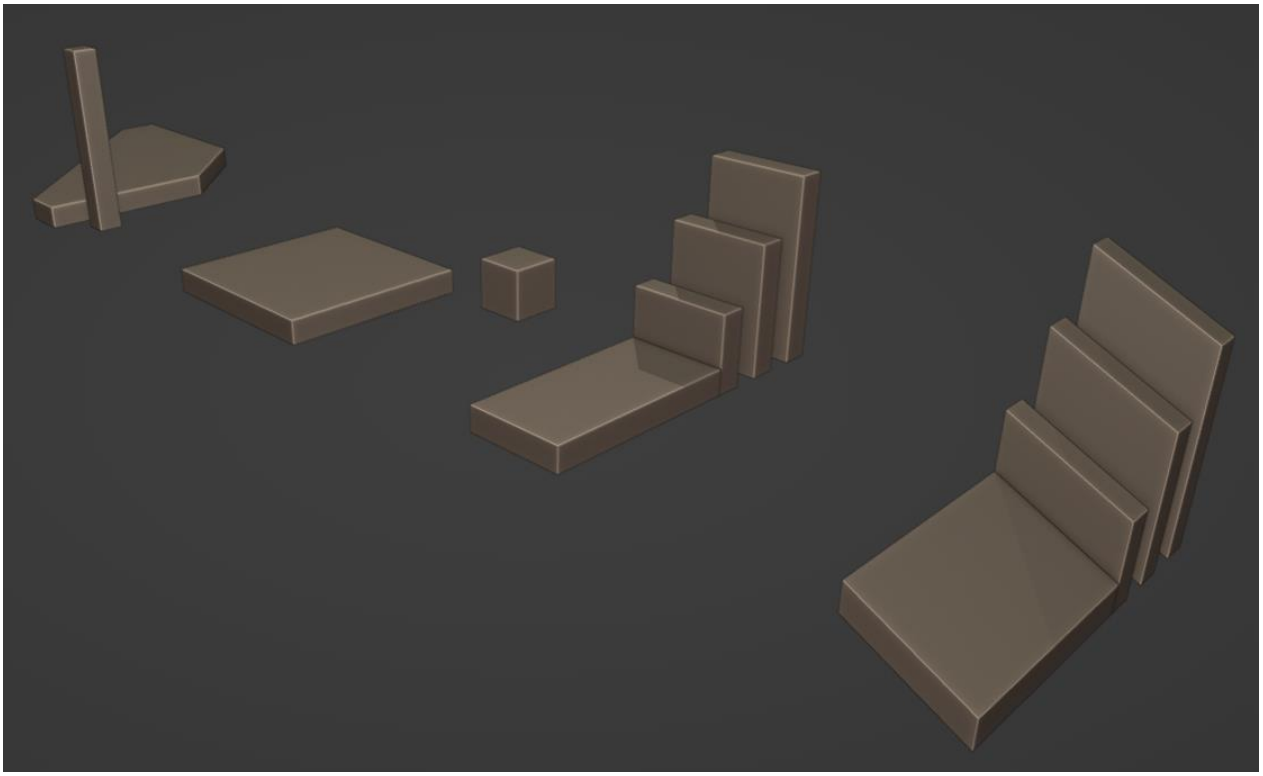
Lékárnička



Dron



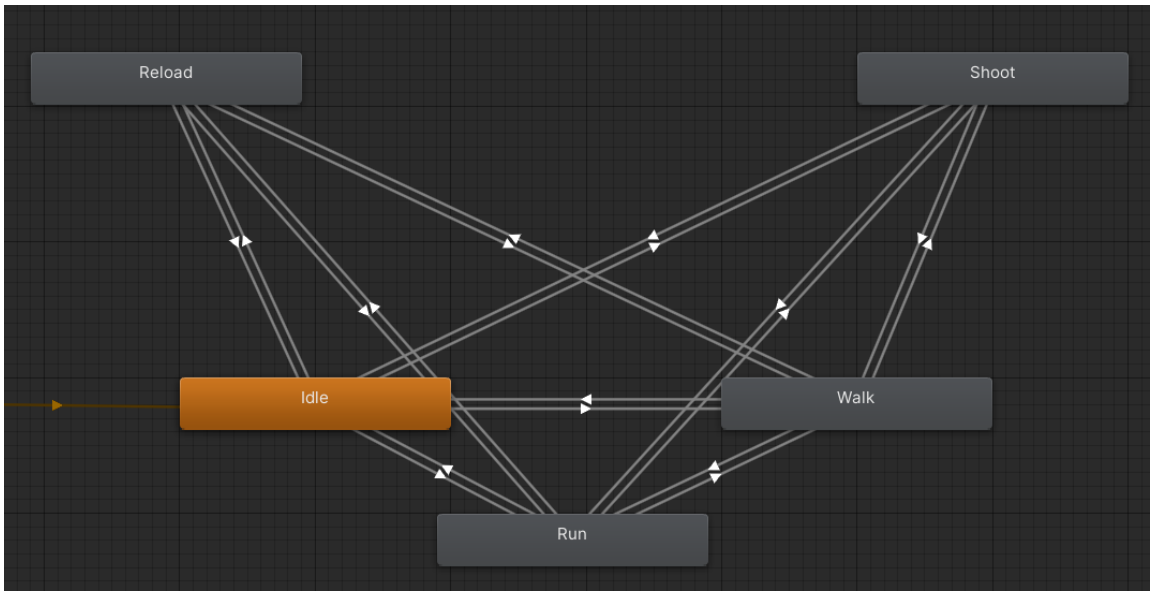
Prostředí



Základní stavební bloky jsem vymodeloval tak, aby byly všechny modulární.

Animace

Když přislo na animace, rozhodl jsem se použít AnimatorOverrideController. Ten mi umožnil vytvořit pouze jeden Animator Controller (viz obrázek níže). Následně bylo možné vytvářet jen již zmíněné AnimatorOverrideControllery, které mi umožnily dosadit do již vytvořené Animator struktury jiné animace.

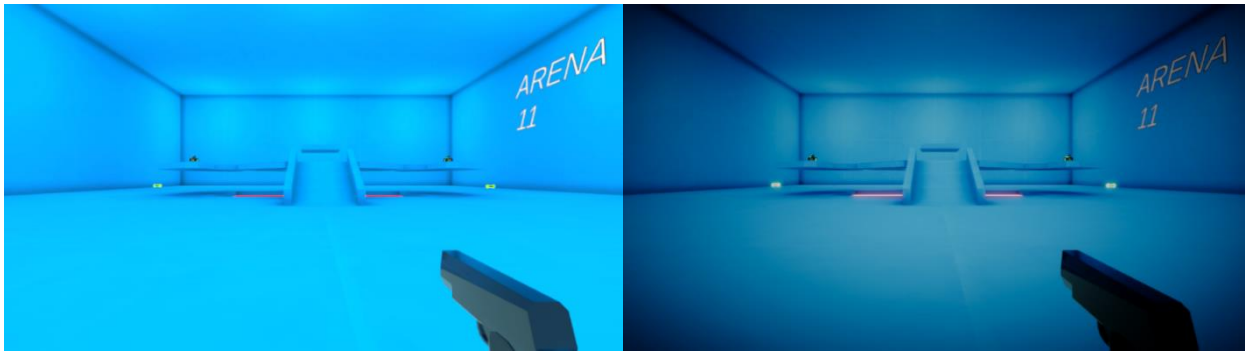


Grafika, osvětlení a Post Processing

Pro můj projekt jsem se rozhodl vybrat URP neboli Universal Render Pipeline, který mi následně dovolil využít i další nástroje, jako je například Visual effect graph.

Všechno osvětlení levelů je tzv. upečené. Tím jsem ušetřili výkon, ale zároveň jsem tím zvětšili celkovou velikost hry, protože se všechno osvětlení uloží do textur.

Hra používá i techniku Post Processingu, neboli přidání speciálních efektů na již vyrenderovaný snímek hry.



Porovnání: scéna bez Post Processingu (vlevo) a s Post Processingem (vpravo).

Používám efekty jako: vignette – ztmavení okraje obrazovky, bloom – efekt přidávající “osvětlení” pro materiály se světlo vyzařujícími vlastnostmi (viz zelené lékárničky nebo červený prvek v obrázcích nahoře) a dalších efektech ovlivňujících celkovou barvu snímku.

Scripty

FPS Controller

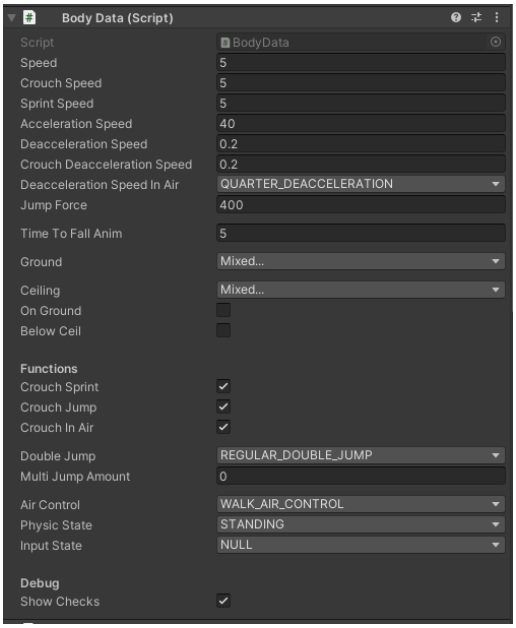
Pohyb hráče je plně fyzický a používá Unity rigidbody fyziku.

```
private void MoveRB()
{

    if (input != Vector3.zero && bodyData.onGround || input != Vector3.zero && bodyData.airControl != BodyData.EAirControl.NONE)
    {
        velocity = new Vector3(input.x * curSpeed, 0, input.y * curSpeed);
        rb.AddRelativeForce(velocity * bodyData.accelerationSpeed, ForceMode.Acceleration);

        Vector2 rbSpeed = Vector2.ClampMagnitude(new Vector2(rb.velocity.x, rb.velocity.z), curSpeed);
        rb.velocity = new Vector3(rbSpeed.x, rb.velocity.y, rbSpeed.y);
    }
    else
    {
        if (bodyData.onGround)
        {
            if (bodyData.inputState == BodyData.EInputState.CROUCH)
            {
                rb.velocity = Vector3.Lerp(rb.velocity, new Vector3(0, rb.velocity.y, 0), bodyData.crouchDeaccelerationSpeed);

            }
            else
            {
                rb.velocity = Vector3.Lerp(rb.velocity, new Vector3(0, rb.velocity.y, 0), bodyData.deaccelerationSpeed);

            }
        }
        else
        {
            rb.velocity = Vector3.Lerp(rb.velocity, new Vector3(0, rb.velocity.y, 0), bodyData.deaccelerationSpeed / (int) bodyData.deaccelerationSpeedInAir);
        }

    }
```

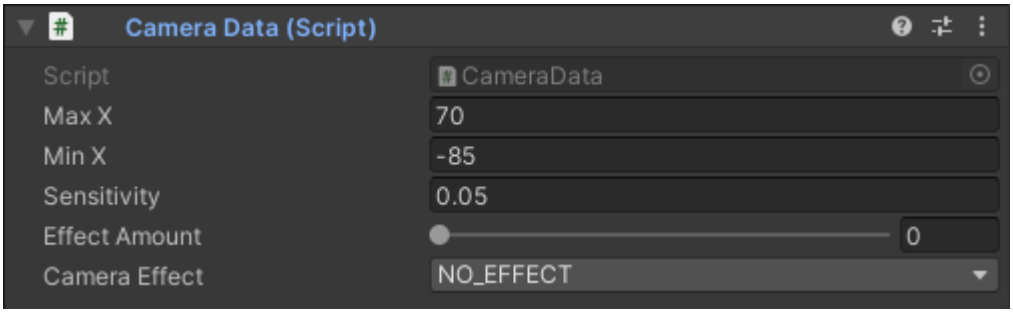
Celý systém se skládá ze čtyř částí:

BodyData.cs, BodyController.cs a CameraData.cs a CameraController.cs

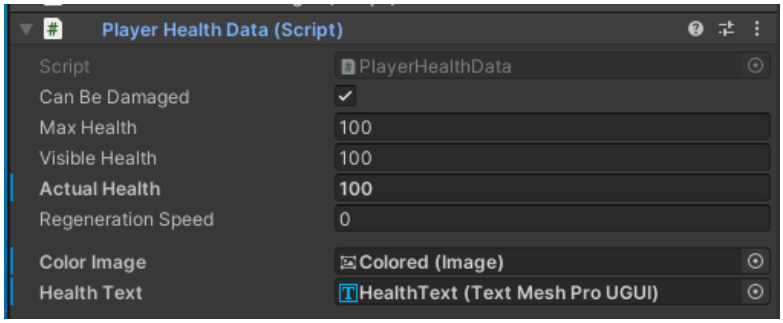
První je script obsahující jen proměnné, které následně BodyController načítá a mění podle stavu, v jakém se nachází.



To samé platí u ovládání kamery (v již zmíněném CameraData a CameraController):



Stejnou strukturu používáme i u životů hráče:



PlayerHealthData.cs obsahuje jen proměnné, které načítáme a používáme v PlayerHealthController.cs.

Tuto strukturu jsem zvolil, protože nám umožňuje upravit jakoukoliv věc bez hledání, na jakém objektu je konkrétní script.

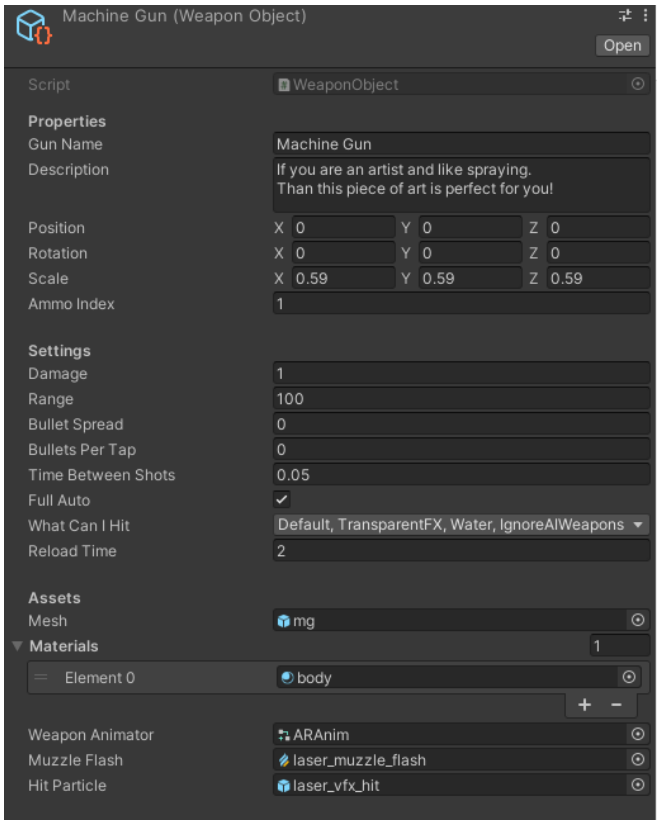
Zbraně

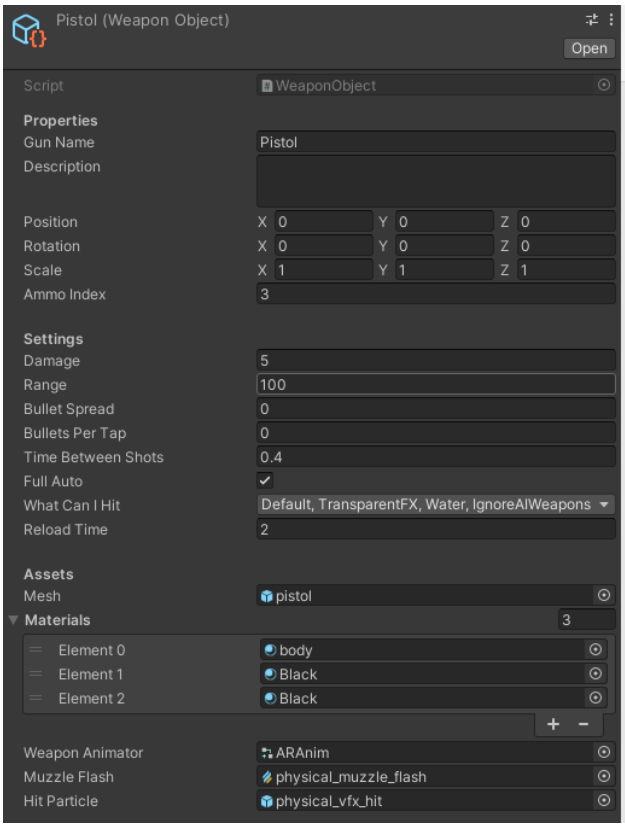
Pro samotné střílení používám techniku Raycasting nebo také Hitscan.

```
if (Physics.Raycast(source.position, direction, out hit, currentWeapon.range, currentWeapon.whatCanIHit))
{
    IHittable ihit = hit.collider.gameObject.GetComponent<IHittable>();

    if (ihit != null)
    {
        for (int i = 0; i < hitEvents.Length; i++)
        {
            if (hit.transform.gameObject.CompareTag(hitEvents[i].tag))
            {
                ihit.GetHit();
                hitEvents[i].events.Invoke();
            }
        }
    }
    Rigidbody rb = hit.collider.gameObject.GetComponent<Rigidbody>();
    if (rb != null)
    {
        rb.AddForce(direction * 2, ForceMode.Impulse);
    }
```

Systém spočívá v tom, že každý výstřel je simulovaný. Vyšle se paprsek, který míří určeným směrem z bodu v prostoru dokud nenarazí na collider.

Jednotlivé zbraně, které si hráč odemkne během hry jsou tzv. ScriptableObjects. Jedná se o script, ze kterého jdou udělat fyzické instance na disk. Na ty lze následně vytvořit referenci v jiném scriptu a načíst data v nich uložené.





Ušetřilo mi to hodně opakování stejného kódu a zlepšilo univerzálnost řešení.

Ovládání

Na ovládání hry je využit nový Input System od Unity.

Umožnil mi mít více univerzální ovládání, protože narozdíl od starého systému je tento upravitelný z jednoho místa oproti decentralizovanému ovládání vstupů.

AI nepřátel

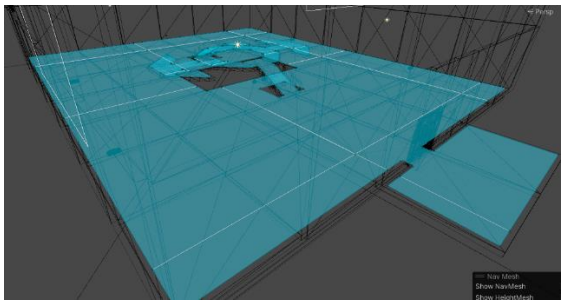
Nepřátelé se spawnují ve vlnách. Při spawnutí se nejdříve objeví Visual effect:

Po spawnutí, nepřítel vyšle raycast směrem k hráči, a pokud ho strefí, začne ho následovat a střílet na něj.

O vlny se stará WaveSystem.cs, který nejdřív procykluje své child objekty a podle nich strukturuje jednotlivé vlny.

```
Zpráva Unity | Počet odkazů: 0
private void Awake()
{


    waves = new List<Wave>();

    for (int i = 0; i < waveObjects.Length; i++)
    {
        //print("wave " + i + "   " + waveObjects[i].transform.childCount);
        waves.Add(new Wave());
        waves[i].enemies = new List<GameObject>();

        for (int j = 0; j < waveObjects[i].transform.childCount; j++)
        {
            //print("enemy " + j);
            waves[i].enemies.Add(waveObjects[i].transform.GetChild(j).gameObject);

        }

    }
```

AI používá NavMeshAgenty, kteří se mohou pohybovat jen na před vygenerovaných plochách NavMesh.



Samotná logika nepřátel je ve scriptu AI_Base.cs, který definuje všechny základní proměnné a metody. Tento script je následně zděděn v dalších scriptech pro jednotlivé typy nepřátel.

Při plánování AI_Base byla univerzálnost hlavním cílem. Z toho důvodu jsem vytvořil i strukturu delegátů Action.

```
Skript Unity | Počet odkazů: 3
public class AI_Base : MonoBehaviour, IHittable
{
    Počet odkazů: 4
    [field: SerializeField] protected Transform target { get; set; }
    Počet odkazů: 2
    [field: SerializeField] protected bool dynamic { get; set; }
    Počet odkazů: 1
    [field: SerializeField] protected AI_Projectile_Weapon weapon { get; set; }

    Počet odkazů: 1
    public bool isAlive { get { return CheckHealth(); } }
    Počet odkazů: 1
    [SerializeField] protected bool isVisible { get { return CheckVisibility(); } }
    Počet odkazů: 1
    [SerializeField] protected bool inRange { get { return CheckRange(); } }

    Počet odkazů: 5
    [field: SerializeField] public float health { get; set; }
    Počet odkazů: 2
    [field: SerializeField] protected int range { get; set; }
    Počet odkazů: 2
    [field: SerializeField] protected int startSpeed { get; set; }
    Počet odkazů: 3
    [field: SerializeField] protected float targetDistance { get; set; }
    Počet odkazů: 1
    [field: SerializeField] protected float maximumDistance { get; set; }
    Počet odkazů: 4
    [field: SerializeField] protected Vector3 targetDirection { get; set; }

    Počet odkazů: 2
    [field: SerializeField] protected Action preUpdateAction { get; set; }
    Počet odkazů: 1
    [field: SerializeField] protected Action postUpdateAction { get; set; }
    Počet odkazů: 1
    [field: SerializeField] protected Action visibleAction { get; set; }
    Počet odkazů: 1
    [field: SerializeField] protected Action inRangeAction { get; set; }
```

```
protected void Update()
{
    preUpdateAction?.Invoke();

    if (isAlive)
    {
        targetDistance = Vector3.Distance(target.position, this.transform.position);
        targetDirection = -(this.transform.position - target.position).normalized;
        Debug.DrawRay(this.transform.position, targetDirection, Color.red,0.2f);
        if (dynamic)
        {
            SetDestination(target);

            if(targetDistance < maximumDistance)
            {
                navmesh.speed = Mathf.Lerp(navmesh.speed, 0, navmesh.acceleration / 10);
            }
            else
            {
                navmesh.speed = Mathf.Lerp(navmesh.speed, startSpeed, navmesh.acceleration / 10);
            }

        }

        if (isVisible)
        {
            visibleAction?.Invoke();

            if (inRange)
            {
                inRangeAction?.Invoke();

                Attack();
            }
        }
    }
    else
    {
        Death();
    }

    postUpdateAction?.Invoke();
}
```

O střílení nepřátel se stará AI_Projectile_Weapon.cs

```
public class AI_Projectile_Weapon : MonoBehaviour
{
    [Header("Spawning Settings")]
    public GameObject spawnPoint;
    public float cooldown;
    [Space]
    [Header("Projectile Properties")]
    public GameObject bulletPrefab;
    public BulletObject bulletObject;
    [Space]
    [Header("Orientation Options")]
    private bool canShoot = true;



    Zpráva Unity | Počet odkazů: 0
    private void Start()
    {
        canShoot = true;
    }

    Počet odkazů: 1
    public void Shoot(Vector3 direction)
    {
        //Instantiate
        //Set bullet object
        if (canShoot)
        {
            canShoot = false;

            GameObject bulletTemp = Instantiate(bulletPrefab, spawnPoint.transform.position, Quaternion.identity);
            Bullet bullet = bulletTemp.GetComponent<Bullet>();
            bullet.bulletObject = bulletObject;
            bullet.Setup(-(spawnPoint.transform.position - direction).normalized);
            Destroy(bulletTemp, 5);

            Invoke("Cooldown", cooldown);
        }

    }
    Počet odkazů: 0
    public void Cooldown()
    {
        canShoot = true;
    }
```

Ten narozdíl od WeaponControlleru nepoužívá Hitscan, ale fyzické projektily.

RPG prvky

Po poražení každého nepřítele hráč dostane určitý počet zkušeností a měny. Po nasbírání dostatečného množství XP se hráči zvedne level. S každým levelem hráč dostane také upgrade point, který společně s měnou může utratit za vylepšení z obchodu.

```
Počet odkazů: 2
private void ResolveXPLevels()
{
    level += (int) xp / xpPerLevel;

    upgradePoints += (int) xp / xpPerLevel;

    xp = xp % xpPerLevel;
}
```

V obchodě si hráč může koupit upgrady jak na zbraně, které zvýší jejich poškození a zvýší rate of fire, tak schopnosti jako EMP Charge nebo Weapon Overcharge. Tyto schopnosti dědí od BaseUpgrade.cs. Toto nám umožňuje rychlé přidání dalších schopností.

EMPCharge

Je schopnost, která každých 20 vteřin dovolí hráči vyslat elektro magnetický výbuch, jenž okamžitě zničí všechny nepřátele v blízkosti. Pole všech nepřátel v blízkosti získáváme díky OverlapSphere.

```
Počet odkazů: 2
public override void Activate()
{
    canPerform = false;
    Invoke("Cooldown", cooldown);


    Collider[] cols = Physics.OverlapSphere(transform.GetChild(0).gameObject.transform.position, radius);

    for (int i = 0; i < cols.Length; i++)
    {
        if (cols[i].CompareTag("AI"))
        {
            cols[i].GetComponent<AI_Base>().health = 0;
        }

    }


    if (post.profile.TryGet<FilmGrain>(out FilmGrain g))
    {
        grain = g;
        grain.intensity.overrideState = true;
        grain.active = true;
        activated = true;
    }
}
```

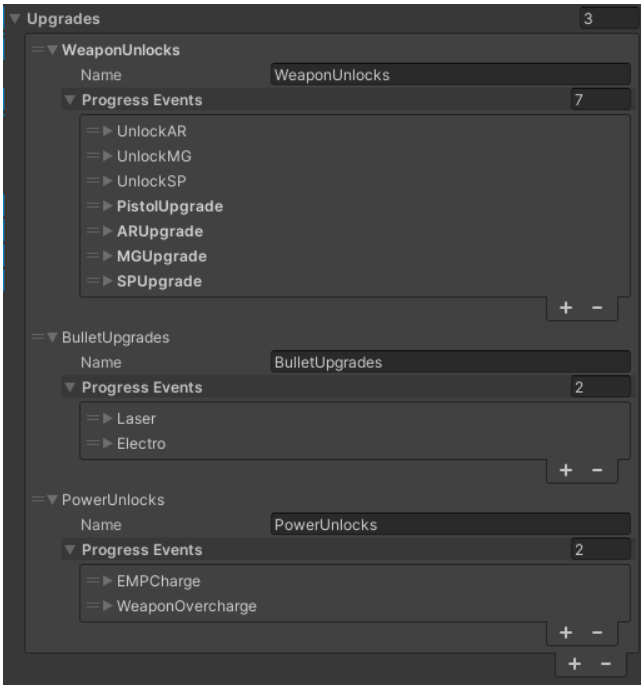
WeaponOvercharge

Je jednoduchá schopnost, která každých 10 vteřin dovolí hráči zvýšit poškození všech držených zbraní na dvojnásobek.

```
Počet odkazů: 2
public override void Activate()
{
    canPerform = false;
    Invoke("Cooldown", cooldown);

    WeaponController.instance.overchargeMultiplier = damageMultiplier;
    print(this.ToString() + " activated");


    Invoke("ResetDamage", length);


    if (post.profile.TryGet<ColorAdjustments>(out ColorAdjustments g))
    {
        color = g;
        color.active = true;
        activated = true;
        Invoke("ResetColor", 2f);
    }
}
```

Manažer

O odemykání nových schopností nebo vylepšení se stará GameProgressManager.cs.



Skládá se z kategorií eventů. Každý event má název, boolean označující jestli je schopnost odemčená, a následně i UnityEvent na metody, které se spustí při splnění daného eventu.

Nepřímou součástí je i SaveProgress.cs.

```
Počet odkazů: 1
public void Save()
{
    Debug.LogWarning("Saved");

    BitArray b = new BitArray(GameProgressManager.instance.GetAllUpgrades());

    int[] output = new int[1];
    b.CopyTo(output, 0);

    PlayerPrefs.SetInt("upgrades", output[0]);
    PlayerPrefs.SetInt("level", ExperienceSystem.instance.level);
    PlayerPrefs.SetInt("xp", ExperienceSystem.instance.xp);
    PlayerPrefs.SetInt("bolts", PlayerCurrency.instance.amount);
    PlayerPrefs.SetInt("upgradePoints", ExperienceSystem.instance.upgradePoints);


    PlayerPrefs.SetInt("ammo1", WeaponController.instance.ammoData.ammoList[0].bullets_left);
    PlayerPrefs.SetInt("ammo2", WeaponController.instance.ammoData.ammoList[1].bullets_left);
    PlayerPrefs.SetInt("ammo3", WeaponController.instance.ammoData.ammoList[2].bullets_left);

    PlayerPrefs.Save();
}
```

```
Počet odkazů: 1
public void Load()
{
    Debug.LogWarning("Loaded");

    BitArray b = new BitArray(new int[] { PlayerPrefs.GetInt("upgrades") });

    bool[] bools = new bool[b.Count];
    b.CopyTo(bools, 0);

    GameProgressManager.instance.SetAllUpgrades(bools);

    ExperienceSystem.instance.level = PlayerPrefs.GetInt("level");
    ExperienceSystem.instance.xp = PlayerPrefs.GetInt("xp");
    PlayerCurrency.instance.amount = PlayerPrefs.GetInt("bolts");
    ExperienceSystem.instance.upgradePoints = PlayerPrefs.GetInt("upgradePoints");

    WeaponController.instance.ammoData.ammoList[0].bullets_left = PlayerPrefs.GetInt("ammo1");
    WeaponController.instance.ammoData.ammoList[1].bullets_left = PlayerPrefs.GetInt("ammo2");
    WeaponController.instance.ammoData.ammoList[2].bullets_left = PlayerPrefs.GetInt("ammo3");

}
```

Ukládám hru pomocí třídy PlayerPrefs, ulehčuje celý proces ukládání hry díky metodám jako SetInt, SetFloat a GetInt a GetFloat. Pro uložení schopností jsem použil místo jednotlivých booleanů pro každou z nich jeden integer. Do něj přes BitArray ukládám stav každého vylepšení a následně při načítání opět přes bitarray je převedu zpět na booleany.

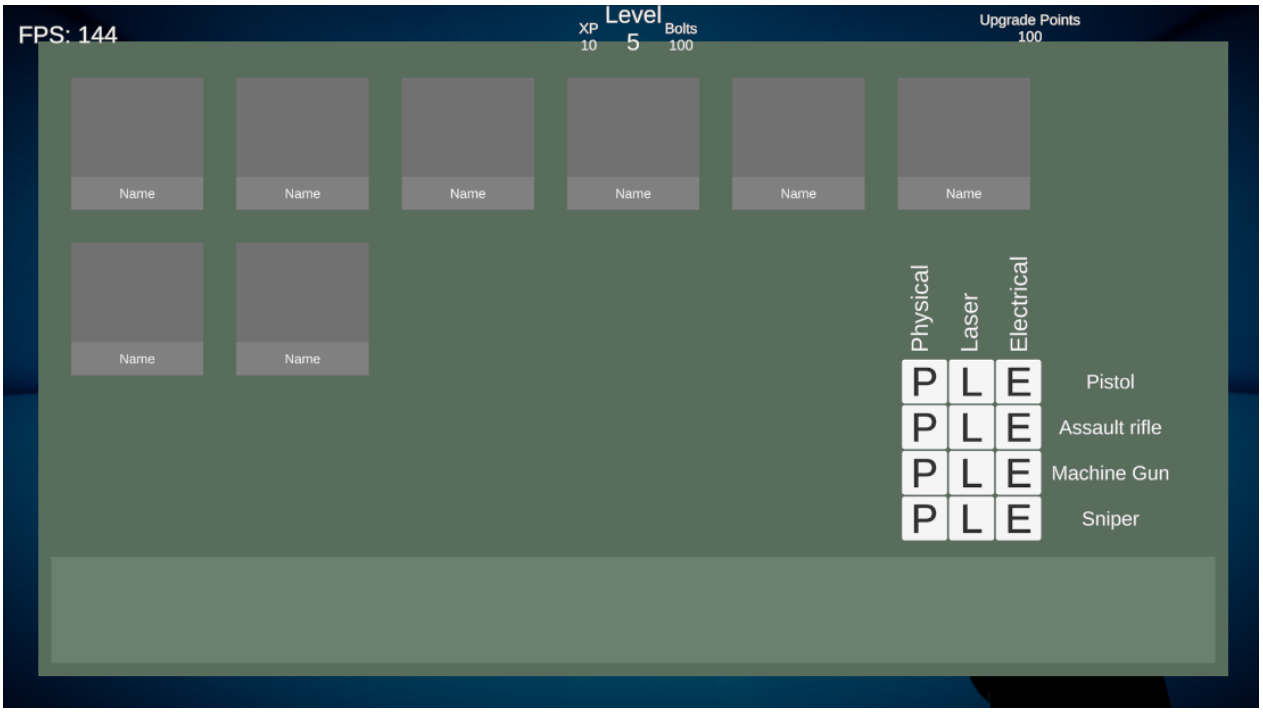
Uživatelské rozhraní

Na uživatelské rozhraní používám program Blender. Důvod je ten, že si s ním rozumím mnohem více, než s jakýmkoliv jiným programem na úpravu fotek.

UI jsem chtěl udělat minimalistické, ale zároveň futuristické, aby sedělo s tématem hry.

HUD elementy

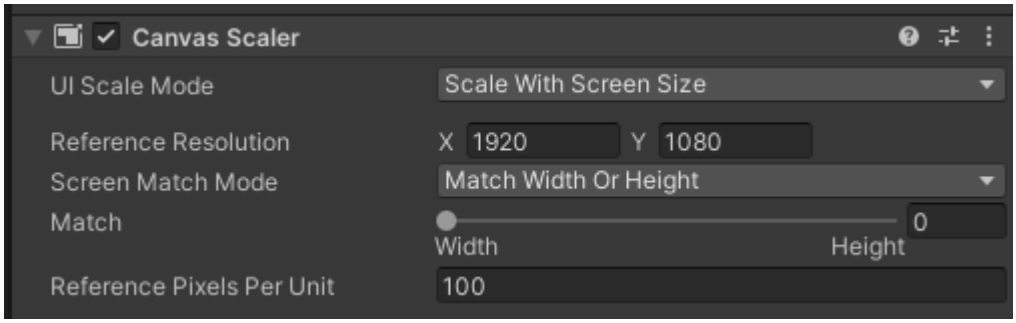
Obchod



Na textové elementy používám balíček TextMeshPro, který mi dává více pokročilých funkcí, například text outline.

Problémy s UI

Při vytvoření prvního buildu jsem zjistil, že UI elementy se špatně škálují a jsou na úplně jiných místech, než by měly být. Po rychlém hledání jsem našel, že se musí v Canvas Scaler komponentu nastavit jiný UI Scale Mode.



Tipy a triky

- Na nepřátele s modrým štítem jsou nejefektivnější electro střely.
- Na heavy černé nepřátele je nejlepší mít na zbrani laser střely.
- Pokud omylem spadnete, nepouštějte WASD, protože se můžete držet zdi a načasovat si tak pád mimo smrtelnou červenou podlahu.
- Hned od začátku hry můžete jít rovnou do druhého levelu pro výzvu, ale tento postup velice nedoporučuji, protože teprve v prvním levelu dostanete další zbraně.
- Doporučená strategie pro nejefektivnější boj je nastavit si jednu zbraň na electro střely, jednu na laser a další na physical.
- Pozor na náboje, například machine gun je na náboje velmi náročný a dokáže je opravdu rychle spotřebovat.
- Kdyz koupíte upgrade/schopnost, dole se ukáže text popisující co daný upgrade/schopnost dělá.
- Změna typu střeliva zbraně je v upgrade okně vpravo dole. Avšak je třeba dávat pozor, aby změna proběhla. K úspěšné změně střel se musí zbraň obnovit, neboli si vybrat jinou, a vrátit se zpět.

Zdroje

Webové stránky

- http://stackoverflow.com/
- https://forum.unity3d.com/
- https://docs.unity3d.com/Manual/index.html

Youtube kanály

- https://www.youtube.com/c/GabrielAguiarProd
- https://www.youtube.com/c/Codeer
- https://www.youtube.com/c/Tarodev
- https://www.youtube.com/channel/UCIWlCE2kt0RXCJLRp8HjhiQ

Závěr

Na SPŠE Ječná jsem šel hlavně z jednoho důvodu: naučit se programovat, abych byl schopný tvořit hry. Již od prvního ročníku, kdy nám bylo řečeno, že lze složit část maturitní zkoušky díky vytvoření hry, byl jsem rozhodnut, že toho využiji.

Jsem rád, že mi tato radost a zájem o vytváření her zůstaly dodnes. Moc se těším na další projekty, pomocí kterých se budu zlepšovat v různých oblastech IT a zejména herního vývoje.