Střední průmyslová škola a Vyšší odborná škola, Liberec, příspěvková organizace

MEDIAPIPE V UNITY

Maturitní práce

Autor	**Matyáš Adamec**
Obor	**Informační technologie**
Vedoucí práce	**Ing. Tomáš Kazda**
Školní rok	**2024/2025**
Počet stran	**34**
Počet slov	**4363**

SPŠSE A VOŠ LIBEREC

Přihláška k maturitní práci

Jméno a příjmení studenta
Adamec, Matyáš

Třída
P4A

Název práce
MediaPipe pro Unity

Školní rok
MP2024/25

Přidělené role

Vedoucí práce

Podpis .....
Kazda, Tomáš

Oponent

Obecná ustanovení	Vypracování a odevzdání práce proběhne v souladu s platnými normami (vyhláška 177/2009 Sb.) a aktuálním dokumentem "Pokyny k vypracování prací" vydaným školou.
	Práce bude hodnocena z hlediska jejího praktického využití, zvládnutí dokumentace po věcné i formální stránce a obhajoby celé práce. Student byl seznámen s kritérii hodnocení maturitní práce.
	Práce bude odevzdána ve dvou stejnopisech vázaných pevnou nebo kroužkovou vazbou.
	Veškeré náklady na MP včetně vyhotovení obou tištěných kopií si student hradí sám.
Licenční ujednání	Ve smyslu § 60 (Školní dílo) autorského zákona č. 121/2000 Sb. poskytuji SPŠSE a VOŠ Liberec výhradní a neomezená práva k využití této mé maturitní práce.
	Bez svolení školy se zdržím jakéhokoliv komerčního využití mé práce.
	Pro výukové účely a prezentaci školy se vzdávám nároku na odměnu za užití díla.

Finanční rozvaha - odhad celkových nákladů

V Kč	Náklady celkem	Hrazené školou
Výrobní	0	0
Na služby	0	0

Jedná se o MP, jejíž vypracování si škola vyžádala? **Ano – Ne**

Podpis studenta (vyjadřuje souhlas s uvedenými údaji a ujednáními)
V Liberci 01.10.2024
Podpis .....

Konzultant
Práci podporuji
Podpis .....

Předmětová komise
Práci doporučuji
Podpis .....

Třídní učitel
Práci doporučuji
Podpis .....

Garant oboru
Práci doporučuji
Podpis .....

Ředitel školy
Práci doporučuji
Podpis .....

Zadání maturitní práce

Název

MediaPipe pro Unity

Předmět

PRG

Téma

Tato maturitní práce se zaměřuje na implementaci a integraci technologie sledování rukou pomocí knihovny MediaPipe do herního enginu Unity. Cílem je vytvořit systém, který bude poskytovat data o poloze rukou v reálném čase. Tato data budou následně využita v Unity pro manipulaci s virtuálními objekty ve scéně. Práce zahrnuje následující kroky:

Použité prostředky

počítač, intel realsense

Cíle práce

1	Nastavení a konfigurace sledování rukou pomocí MediaPipe
2	Vytvoření systému pro poskytování dat o poloze rukou
3	Vývoj herní scény v Unity, kde ruce slouží pro manipulaci s objekty

Osnova práce

2	Nastavení a konfigurace MediaPipe Hand Tracking
3	Integrace s Unity
4	Vývoj herní scény v Unity
1	Knihovna MediaPipe

Anotace

Tato práce se zaměřuje na propojení hloubkové kamery Intel RealSense a knihovny MediaPipe za účelem vytvoření nových možností ovládání a zpracování pohybu v Unity. Kombinace těchto technologií má široké potenciální využití ve vývoji počítačových her a rozšířené realitě, kdy umožňuje prohloubený a intenzivnější zážitek.

Práce se zabývá kromě vytvoření samotné technologie i možnostmi její integrace tak, aby umožňovala precizní sledování pohybu rukou ve 3D prostoru a interakci s herním prostředím.

Summary

This work focuses on integrating the Intel RealSense depth camera with the MediaPipe framework to enable additional control possibilities in Unity. The combination of these technologies has several potential applications in computer game creation and augmented reality, allowing for a more immersive and intense experience.

In addition to developing the technology, this thesis investigates how it can be incorporated to provide exact tracking of hand movement in 3D space and interaction with the gaming world.

Čestné prohlášení

Prohlašuji, že jsem předkládanou maturitní práci vypracoval sám a uvedl jsem veškerou použitou literaturu a bibliografické citace.

V Liberci dne 12.03.2025 .....

Matyáš Adamec

Obsah

Úvod ... 1
1 Cíl práce ... 2
1.1 Motivace projektu ... 2
2 Použité technologie ... 3
2.1 Unity ... 3
2.2 Intel Realsense ... 3
2.3 Mediapipe ... 3
2.4 C# ... 3
2.5 Blender ... 3
2.6 Java ... 4
2.7 Python ... 4
2.8 SolidWorks ... 4
3 Hardwarová konfigurace ... 5
3.1 Radxa Rock 5B+ ... 5
3.2 Intel Realsense ... 5
3.3 Projektor ... 5
4 Příprava a konfigurace vývojového prostředí ... 6
5 Nastavení podpory RealSense pro Android ... 7
5.1 Kompilace RealSense SDK pro Android ... 7
5.2 Úprava RealSense knihovny pro Unity ... 8
5.3 Využití AndroidJavaObject a AndroidJavaClass ... 8
6 Integrace Mediapipe do Unity ... 10
6.1 Build knihovny MediaPipe ... 10
6.2 Vazba mezi Unity a nativním kódem ... 11
7 Přesná detekce polohy rukou v prostoru ... 12
7.1 Získání souřadnic landmarků pomocí Unity ... 12
7.2 Převod souřadnic zápěstí na 3D bod pomocí RealSense ... 13
7.3 Transformace souřadnic landmarků do Unity ... 14
8 Sledování a vyhodnocování pohybu rukou ... 15
9 Implementace herní scény v Unity ... 16
9.1 Koncept hry ve stylu BeatSaber ... 16
9.2 Herní menu ... 16

9.3 Hlavní herní scéna ... 17
9.3.1 Generování letících „bloků“ ... 18
9.3.2 Využité kolizí pro detekci zásahu rukou ... 20
9.3.3 Destrukce bloků a odezva ... 20
10 Možnosti rozšíření a budoucí vývoj ... 22
Závěr ... 23
Seznam zkratek a odborných výrazů ... 24
Seznam obrázků ... 25
Použité zdroje ... 26
A. Seznam přiložených souborů ... I

Úvod

K této práci jsem se dostal prostřednictvím startupu nullspaces s.r.o., díky kterému jsem získal jedinečnou příležitost zapojit se do vývoje systému využívajícího MediaPipe a RealSense v prostředí Unity. Cílem projektu bylo vytvořit řešení, které umožní ovládání digitálního prostředí zcela bez tradičních ovladačů či dotykových obrazovek. Namísto toho systém pracuje s přesným sledováním 3D polohy rukou pomocí technologie Intel RealSense a s využitím knihovny MediaPipe, která v reálném čase detekuje klíčové body ruky z obrazového signálu.

Jelikož jsem se doposud nikdy nevěnoval úlohám spojeným se strojovým učením a zpracováním obrazu, byl jsem fascinován samotným procesem učení systému a získáváním dat, která se následně využívají k optimalizaci a zlepšení přesnosti sledování pohybů. Celý proces mě zaujal především tím, jak se moderní technologie dají propojit a využít pro tvorbu robustních řešení, která dokážou posunout hranice současných interaktivních technologií.

Práce na tomto projektu mi tak nabídla nejen cenné zkušenosti z oblasti programování a integrace pokročilých senzorů, ale také hlubší vhled do světa strojového učení a zpracování obrazu.

1 Cíl práce

Cílem této práce je vytvořit základní kostru technologie snímání pohybu, která bude sloužit jako výchozí platforma pro řadu inovativních projektů. Projekt se zaměřuje na propojení knihovny Mediapipe a technologie Intel RealSense v herním enginu Unity, přičemž výsledná aplikace poběží na platformě Android.

Hlavním záměrem je využít schopnosti hloubkové kamery RealSense pro 3D analýzu prostoru a ovládání aplikace rukama. Mediapipe je pak použit k zpracování obrazu v reálném čase pomocí strojového učení, což umožňuje přesné vyhodnocení polohy rukou.

1.1 Motivace projektu

Tento projekt byl zahájen s cílem reagovat na rostoucí potřebu interaktivních technologií, které umožňují přirozené ovládání digitálního prostředí. V době, kdy ovládání pomocí gest a dotyků získává na popularitě, bylo důležité vyvinout systém, který by umožňoval přesné sledování pohybu rukou a jejich integraci do herních a dalších aplikací. Projekt byl inspirován snahou o vytvoření intuitivního ovládání, které by překonalo tradiční způsoby ovládání pomocí tlačítek nebo dotykových obrazovek.

2 Použité technologie

V této kapitole budou blíže rozvedeny technologie využívané v tomto projektu a důvody, proč byly k vypracování projektu vybrány právě ony.

2.1 Unity

Unity je herní engine, který byl v tomto projektu použit pro vývoj celé aplikace. Nabízí nástroje pro vývoj ve 2D i 3D. Obsahuje fyzikální, vykreslovací i zvukový engine, jež umožňují tvorby a simulace reálného prostředí.

2.2 Intel Realsense

Intel RealSense je hloubková kamera, která byla v projektu použita k přesnému sledování pohybu rukou v prostoru. Na rozdíl od běžných kamer, které poskytují pouze 2D obraz, RealSense dokáže určovat 3D souřadnice objektů v reálném čase. V projektu byla použita hlavně k získávání polohy zápěstí, které slouží jako referenční bod pro celou ruku. Díky tomu lze sledovat pohyb rukou bez nutnosti použití ovladačů.

2.3 Mediapipe

MediaPipe je framework pro zpracování obrazu pomocí strojového učení. V tomto projektu se používá pro vyhodnocování lokace rukou. Na základě obrazu z kamery dokáže detekovat klíčové body ruky a určit jejich polohu.

2.4 C#

C# je hlavní programovací jazyk používaný v Unity, a proto byl využit i v tomto projektu. Jedná se o objektově orientovaný jazyk, který umožňuje efektivní správu herních objektů, detekci kolizí a propojení hry s dalšími technologiemi, jako je sledování rukou a zpracování pohybu.

2.5 Blender

Blender byl použit pro tvorbu 3D modelů herních bloků. Tento software umožňuje modelování, texturování a export objektů do formátu, který lze snadno importovat do Unity. Modely bloků byly vytvořeny tak, aby odpovídaly vizuálnímu stylu hry a zároveň byly optimalizované pro plynulý běh aplikace.

2.6 Java

Java byla využita k úpravám RealSense SDK na Androidu. Některé části knihovny bylo nutné přizpůsobit, aby správně fungovaly v Unity. Pomocí Javy byly například upraveny metody pro zpracování hloubkových dat a jejich převod do podoby, kterou Unity dokáže interpretovat.

2.7 Python

Python je vysokoúrovňový programovací jazyk, který je široce využívaný pro zpracování dat a automatizaci úloh. V tomto projektu byl použit k analýze audia pomocí knihovny pydub, která audio rozděluje na segmenty a detekuje okamžiky vrcholů, jež jsou následně využity ke generaci a synchronizaci herních bloků s rytmem hudby.

2.8 SolidWorks

SolidWorks 2024 je CAD software využívaný pro návrh a tvorbu 3D modelů. V tomto projektu byl SolidWorks použit při konstrukci krabičky, do které byly integrovány zařízení Radxa Rock 5B+ a Intel RealSense.

Obrázek 1 Konstrukce krabičky

3 Hardwarová konfigurace

Výběr hardwarové konfigurace v tomto projektu byl proveden s důrazem na výpočetní výkon, kompatibilitu s vybranými technologiemi a potenciál pro budoucí rozšíření. Klíčovými komponentami konfigurace jsou vývojová deska Radxa Rock 5B+, poskytující dostatečný výpočetní výkon, a hloubková kamera Intel RealSense, umožňující precizní sledování rukou ve 3D prostoru.

3.1 Radxa Rock 5B+

Pro tento projekt byla vybrána vývojová deska Radxa Rock 5B+, a to především díky svému výkonnému NPU a přítomnosti dvou konektorů pro Coral akcelerátory. Tyto akcelerátory sice nejsou v aktuální implementaci využívány, avšak jejich přítomnost umožňuje případné rozšíření funkcionality v budoucnu. (1)

Jako operační systém byl zvolen Android, a to především kvůli široké podpoře od Unity, což výrazně usnadňuje celkový vývoj, kompatibilitu i nasazení aplikace. Dalším důvodem byla také kompatibilita s Intel RealSense, přičemž přímo na oficiálních stránkách společnosti Intel je k dispozici ukázková implementace pro Android. Významnou výhodou byla také oficiální podpora Androidu ze strany výrobce Radxa, což zajišťuje stabilitu systému a dostupnost potřebných ovladačů. (2)

Aby bylo možné správně využívat Intel RealSense, bylo nezbytné provést root zařízení. Rootování umožňuje přístup k pokročilým funkcím systému, které jsou nutné pro práci s RealSense kamerou a jejími specifickými funkcemi. (3)

3.2 Intel Realsense

Intel RealSense je pokročilá hloubková kamera, která umožňuje přesné snímání 3D prostoru v reálném čase. Díky kombinaci RGB kamery, infračerveného projektoru a hloubkového senzoru dokáže získávat podrobná 3D data o okolním prostředí.

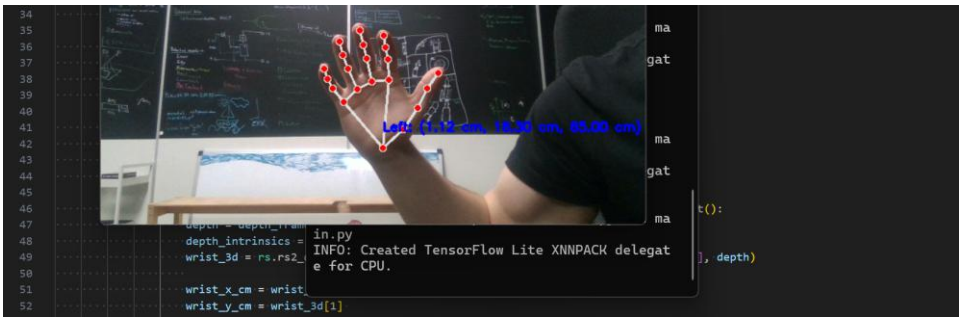
V rámci tohoto projektu je RealSense využívána ke sledování rukou ve trojrozměrném prostoru. Na rozdíl od běžných 2D kamer umožňuje přesně určit nejen polohu rukou v rovině, ale i jejich hloubku (Z-ovou souřadnici), což je klíčové pro interakci ve virtuálním prostředí. (4)

3.3 Projektor

Projektor je v tomto projektu využit k promítání herní scény na zeď, čímž vytváří interaktivní prostředí, ve kterém hráč fyzicky interaguje s vizuálními prvky pouze pomocí pohybů rukou. Díky této technologii je možné odstranit potřebu monitoru či VR headsetu a umožnit hráči přirozenou a volnou interakci s projekcí ve skutečném prostoru.

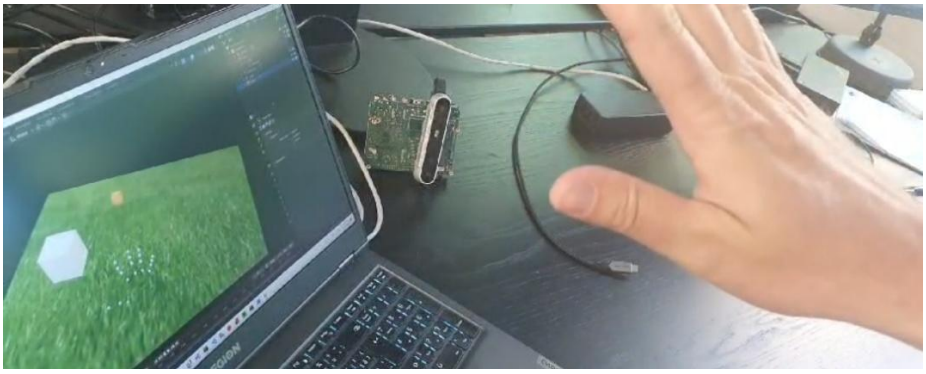
4 Příprava a konfigurace vývojového prostředí

V rámci přípravy a konfigurace vývojového prostředí jsem nejprve strávil mnoho hodin studiem dokumentace a experimentováním s různými pokusnými scripty. S MediaPipe a RealSense jsem se musel nejprve naučit pracovat prostřednictvím Pythonu, abych pochopil, jak tyto technologie fungují, a abych se vyhnul mnohem složitějším problémům při jejich implementaci v Unity.



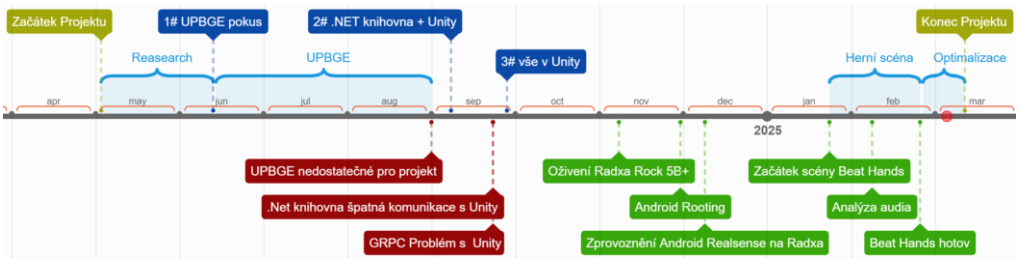
Obrázek 2 Testování MediaPipe a Intel RealSense v Pythonu

Následně byl uskutečněn pokus o integraci MediaPipe a RealSense do UPBGE, což je fork Blenderu s herním enginem. Tento pokus se však setkal s řadou problémů, protože UPBGE bylo výrazně pozadu a nedosahovalo úrovně, kterou nabízí Unity.



Obrázek 3 MediaPipe a RealSense v Unity

Ačkoli jsem s Unity již dříve pracoval, musel jsem se s tím učit prakticky od začátku, abych se seznámil s aktuálními postupy a technikami. První verze mého řešení spočívaly ve vytváření vlastních .NET knihoven s cílem extrahovat pouze souřadnice landmarků z RealSense a MediaPipe a následně je používat v Unity. Tento přístup se však ukázal jako příliš komplikovaný, a proto jsem nakonec integroval obě technologie přímo do prostředí Unity, což značně zjednodušilo další vývoj.



Obrázek 4 Časová osa projektu

5 Nastavení podpory RealSense pro Android

Aby bylo možné využívat RealSense SDK v Unity na platformě Android, bylo nezbytné provést několik úprav a přizpůsobení knihovny. Tyto úpravy zahrnovaly kompilaci RealSense SDK pro architekturu Android, modifikaci vybraných tříd a jejich integraci do Unity.

5.1 Kompilace RealSense SDK pro Android

RealSense SDK nebylo možné použít v Unity přímo ve výchozí podobě, a proto bylo nutné sestavit knihovnu ve formátu AAR (Android Archive). Tento formát umožňuje integraci nativního kódu do Unity jako Android plugin.

Nejprve je nutné stáhnout zdrojový kód knihovny RealSense. K tomu slouží klonování oficiálního repozitáře pomocí příkazu:

```
git clone https://github.com/IntelRealSense/librealsense.git
cd librealsense/wrappers/android
```

Následně je potřeba připravit prostředí pro kompilaci. RealSense SDK využívá Gradle, což je nástroj pro správu buildů v Android projektech. Před samotnou kompilací je nutné nastavit správnou verzi Javy, protože starší nebo novější verze mohou způsobit chyby. Definování cesty k JDK 17 se provede následujícím příkazem:

```
gradle wrapper -
Dorg.gradle.java.home="C:\Program\Files\Java\jdk-17"
```

Po nastavení prostředí lze přistoupit k sestavení knihovny. Pro vygenerování AAR knihovny je potřeba spustit následující příkaz:

```
gradlew assembleRelease -Dorg.gradle.java.home="C:\Program
Files\Java\jdk-17"
```

Pokud kompilace proběhne úspěšně, vygenerovaná knihovna se uloží do složky:

```
<librealsense_root_dir>/wrappers/android/librealsense/build/
outputs/aar
```

Knihovna RealSense byla sestavena tímto způsobem, což umožňuje její následné použití s Androidem. (5)

5.2 Úprava RealSense knihovny pro Unity

Kromě samotné kompilace bylo nutné provést úpravy vybraných tříd java knihovny, protože knihovna je vytvořena primárně pro Android Studio, a neobsahuje tedy potřebné metody pro získání dat v Unity (6). Tyto úpravy se týkaly zejména získávání 3D souřadnic bodů, zpracování hloubkových snímků a převodu pixelových souřadnic na souřadnice prostorové.

Například třída Intrinsic byla upravena tak, aby poskytovala informace o interních parametrech kamery, jako jsou ohniskové vzdálenosti a střed obrazu. Tyto hodnoty jsou klíčové pro přesný převod pixelových souřadnic do 3D prostoru. Níže je uvedena implementace třídy Intrinsic, která byla do kódu přidána v Javě:

```
package com.intel.realsense.librealsense;
public class Intrinsic {
    public float getPpx() { return mPpx; }
    public float getPpy() { return mPpy; }
    public float getFx() { return mFx; }
    public float getFy() { return mFy; }
}
```

Podobné úpravy byly provedeny také v třídách Frame a Point_3D, které nyní umožňují efektivní extrakci dat ze snímků a získávání 3D souřadnic bodů. Díky těmto změnám lze data z RealSense knihovny snadno přenášet do Unity, kde jsou využívána k přesnému sledování a vykreslení objektů v 3D prostoru. Tento postup zjednodušil integraci hloubkových dat a přispěl k vytvoření robustního systému pro interakci v reálném čase.

5.3 Využití AndroidJavaObject a AndroidJavaClass

Pro integraci knihovny RealSense SDK do Unity na platformě Android je nutné využít třídy AndroidJavaObject a AndroidJavaClass. Tyto třídy umožňují interakci mezi Unity a nativním kódem Androidu, což je klíčové pro volání metod z AAR knihovny přímo v C#.

V Unity je potřeba načíst snímky z RealSense kamery pomocí RealSense pipeline. To se dělá tak, že se pomocí AndroidJavaObject zavolá metoda "waitForFrames", která vrací aktuální snímky z kamery. Poté se z těchto snímků získá barevný snímek a hloubkový snímek, přičemž barevný snímek se převede na Texture2D, který lze zobrazit ve hře. (7)

```
public Texture2D GetCurrentRealSenseTexture()
{
  Frames =
_pipeline.Call<AndroidJavaObject>("waitForFrames");
  if (frames != null)
  {
    colorFrame = frames.Call<AndroidJavaObject>("first",
_colorEnum);
    depthFrame = frames.Call<AndroidJavaObject>("first",
_depthEnum);
    frames.Call("close");
    if (depthFrame != null)
    {
      byte[] mydata = depthFrame.Call<byte[]>("getMyData");
      _depthTexture.LoadRawTextureData(mydata);
      _depthTexture.Apply();
      depthFrame.Call("close");
    }
    if (colorFrame != null)
    {
      data = colorFrame.Call<byte[]>("getData");
      for (int y = 0; y < _colorHeight; y++)
      {
          Buffer.BlockCopy(data, y * stride, flippedData,
(_colorHeight - y - 1) * stride, stride);
      }
      _colorTexture.LoadRawTextureData(flippedData);
      _colorTexture.Apply();
      colorFrame.Call("close");
      return _colorTexture;
    }
  }
}
```

6 Integrace Mediapipe do Unity

Integrace MediaPipe do Unity byla provedena na základě projektu MediaPipeForUnity od Homulera. Cílem bylo umožnit využití MediaPipe pro detekci hand landmarks přímo v prostředí Unity.

6.1 Build knihovny MediaPipe

Nejprve byl získán zdrojový kód projektu MediaPipeForUnity od Homulera, který byl následně klonován z oficiálního repozitáře. Pro sestavení knihovny bylo nutné připravit odpovídající vývojové prostředí, přičemž byla nainstalována potřebná softwarová vybavení, jako jsou Docker, Android Studio se správně nakonfigurovaným SDK a NDK, Python (verze ≥ 3.9 a < 3.13) a Bazelisk.

V první fázi byl vytvořen pracovní Docker image, ve kterém byla zajištěna kompatibilita se systémem a předchystána prostředí pro kompilaci. Tento krok byl proveden příkazem:

```
Docker build --build-arg UID=$(id -u) -t
mediapipe_unity:latest . -f docker/linux/x86_64/Dockerfile
```

Poté byl spuštěn Docker container, do kterého byly namapovány složky s Assets a Packages. Tím byla zajištěna oboustranná synchronizace souborů mezi hostitelským počítačem a prostředím uvnitř containeru.

Následně byl v Docker containeru spuštěn build skript, který sestavil knihovny pro dvě platformy současně – desktopovou verzi a verzi pro Android. Příkaz využíval parametry určující build pro desktop s podporou CPU a zároveň pro Android (architektura arm64). K tomu byl použit příkaz:

```
python build.py build --desktop cpu --opencv cmake --android
arm64 -v
```

Během tohoto procesu byly automaticky stáhnuty a integrovány potřebné závislosti, jako například OpenCV, a byla provedena optimalizace sestavení. Po úspěšném dokončení buildu byly vygenerované knihovny a soubory umístěny do příslušných složek, což umožnilo jejich následnou integraci do projektu v Unity. Tento postup tak zajistil, že bylo možné využít MediaPipe jak pro desktopové aplikace, tak i jako nativní komponentu pro Android, což umožnilo flexibilní a efektivní nasazení v rámci projektu. (8)

6.2 Vazba mezi Unity a nativním kódem

Pro komunikaci mezi Unity (C#) a nativním kódem MediaPipe, který je napsán v C/C++, byl využit mechanismus DllImport. Díky této metodě mohou být nativní funkce volány přímo z Unity. Například při získávání normalizovaných souřadnic landmarků je využito následující volání:

```
[DllImport(MediaPipeLibrary, ExactSpelling = true)]
public static extern MpReturnCode
mp_Packet__GetNormalizedLandmarksVector(
      IntPtr packet,
      out NativeNormalizedLandmarksArray value
);
```

Tímto způsobem jsou výsledky výpočtu z MediaPipe (např. pozice landmarků) předávány z nativní knihovny do Unity, kde jsou dále zpracovávány a používány k vykreslení a vyhodnocení pohybu rukou. (9)

7 Přesná detekce polohy rukou v prostoru

V této kapitole je popsán proces získávání souřadnic rukou a jejich transformace do Unity. K tomu je využita kombinace MediaPipe a RealSense, které umožňují detekci landmarků ruky v obraze a určení jejich skutečné polohy ve 3D prostoru. Celý postup zahrnuje několik klíčových kroků:

1. získání landmarků pomocí MediaPipe
2. převod souřadnic zápěstí na 3D bod pomocí RealSense
3. transformace landmarků do souřadnic Unity

7.1 Získání souřadnic landmarků pomocí Unity

Prvním krokem celého procesu je vložení obrazu rukou do MediaPipe, které provede analýzu snímku a vrátí souřadnice jednotlivých landmarků. Tyto souřadnice jsou normalizovány do intervalu 0 až 1, což znamená, že nejsou přímo spojeny s rozlišením obrazu.

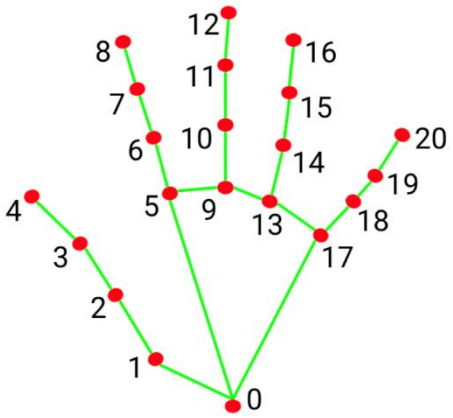
Jelikož jsou landmarky reprezentovány pouze ve 2D, je nutné je převést do souřadnicového systému snímku podle rozlišení kamery. Tento krok zajistí, že landmarky odpovídají skutečným pixelovým souřadnicím na obrazu. Nejdůležitější je landmark zápěstí (index 0), který slouží jako referenční bod pro celou ruku.

V Unity je pro načtení obrazu a detekci landmarků použit následující kód:

```
var texture =
_realsenseManager.GetCurrentRealSenseTexture();
req = textureFrame.ReadTextureAsync(texture, true, false);
yield return waitUntilReqDone;
image = textureFrame.BuildCPUImage();
textureFrame.Release();
taskApi.DetectAsync(image, GetCurrentTimestampMillisec());
```

Po zpracování obrazu se v metodě OnHandLandmarkDetectionOutput vykreslí získané landmarky:

```
private void
OnHandLandmarkDetectionOutput(HandLandmarkerResult result,
Image image, long timestamp)
{
    // vykreslení landmarků
}
```



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Obrázek 5 21 klíčových bodů ruky (10)

7.2 Převod souřadnic zápěstí na 3D bod pomocí RealSense

Jakmile jsou získány souřadnice landmarků, je nutné zjistit skutečnou 3D pozici zápěstí v prostoru. Protože MediaPipe neposkytuje hloubková data, je nutné použít RealSense, která umožňuje rekonstrukci hloubky na základě hloubkového obrazu.

Souřadnice zápěstí jsou nejprve převedeny na pixelové souřadnice snímku a následně je pomocí RealSense určena jeho hloubková hodnota (Z souřadnice).

Následující metoda získá souřadnice zápěstí v prostoru:

```
private (float x, float y, float z)
getPositionFromDepthTexture((float x, float y, float z)
wrist, bool isRight)
{
    var depthTexture = realSenseManager.GetDepthTexture();
    var colorIntrin = realSenseManager.GetIntrin();
    // pixel coordinates of the wrist
    var x = 1280 - (int)(wrist.x * 1280);
    var y = 720 - (int)(wrist.y * 720);
    if (0 <= x && x < colorIntrin.width && 0 <= y && y <
colorIntrin.height)
    {
        var vx = (x - colorIntrin.ppx) / colorIntrin.fx;
        var vy = (y - colorIntrin.ppy) / colorIntrin.fy;
        // Get the raw texture data
        byte[] rawData = depthTexture.GetRawTextureData();
        // Calculate the index of the pixel
        int index = (int)y * depthTexture.width * 2 + (int)x
* 2;
```

```
        // Convert the byte data to ushort
        ushort pixelValue = BitConverter.ToUInt16(rawData,
index);
        var vz = pixelValue * 0.001f;
        return (vx, vy, vz);
    }
}
```

7.3 Transformace souřadnic landmarků do Unity

Jakmile jsou k dispozici souřadnice landmarků z MediaPipe a 3D souřadnice zápěstí z RealSense, je možné provést transformaci ruky do Unity.

Prvním krokem je nastavení zápěstí jako počátečního bodu, což znamená, že v Unity bude vždy na (0,0,0). To se provede odečtením souřadnic zápěstí od všech ostatních landmarků. Tento krok zajistí, že ruka se nebude deformovat a zůstane správně zarovnaná.

Poté jsou upravené landmarky přesunuty na správnou pozici ve světových souřadnicích Unity tak, že se ke každému bodu přičte pozici zápěstí ve 3D prostoru.

Následující metoda provádí tento výpočet:

```
private Vector3 AdjustLocation(
    NormalizedLandmark normalizedLandmark,
    (float x, float y, float z) wristLandMark,
    (float x, float y, float z) realsensePosition
)
{
    Vector3 position = new Vector3(
        (normalizedLandmark.x - wristLandMark.x) +
realsensePosition.x,
        (normalizedLandmark.y - wristLandMark.y) +
realsensePosition.y,
        (1 - (normalizedLandmark.z - wristLandMark.z)) +
realsensePosition.z );
    return position;
}
```

8 Sledování a vyhodnocování pohybu rukou

Během vývoje asynchronního zpracování dat byly zaznamenány různé problémy, které výrazně ovlivňovaly výkon aplikace. Aby bylo možné přesně určit, která část zpracování bere nejvíce času, byl kód přepsán na synchronní variantu a rozložen do jednotlivých segmentů. Každý segment byl časován zvlášť, což umožnilo zjistit dobu trvání jednotlivých úseků. Byly měřeny následující segmenty:

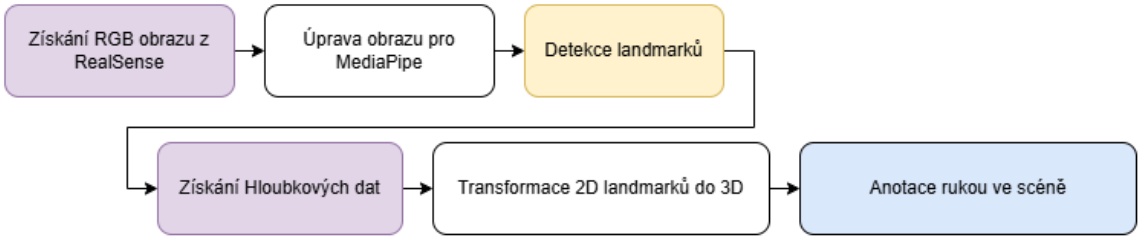
1. Získání RGB obrazu z RealSense – přibližně 32 ms
2. Úprava obrazu pro MediaPipe – přibližně 0 ms
3. Detekce landmarků – přibližně 14 ms
4. Získání hloubkových dat – přibližně 32 ms
5. Transformace 2D landmarků do 3D – přibližně 0 ms
6. Anotace rukou ve scéně – přibližně 0 ms

Celková doba zpracování tak byla přibližně 78 ms, což odpovídalo pouze 13 fps, a to i přes očekávaných 22 fps (46 ms na snímek). Pro identifikaci problému byl kód rozložen na tyto segmenty a byl časován každý úsek samostatně. Bylo zjištěno, že problém spočíval v segmentu získávání hloubkových dat. Při práci s kamerou RealSense byl totiž místo správného allokování textury zapisován celý objekt, což výrazně prodlužovalo dobu zpracování snímků. Dále bylo zjištěno, že se čekalo na získání snímku na dvou místech, což vedlo k poklesu snímkové frekvence z očekávaných 22 fps na 13 fps.

Po úpravách v segmentu získávání hloubkových dat, která snížila dobu zpracování z 32 ms na přibližně 1,5 ms, byl celkový čas zpracování snímku zkrácen na cca 47,5 ms, což odpovídá přibližně 22 fps. Celkový výkon a plynulost aplikace tak byly výrazně zlepšeny.

Celý problém byl odhalen a úspěšně vyřešen díky podrobnému časovému rozložení kódu a přepsání asynchronního zpracování na synchronní variantu. Tím bylo umožněno přesně určit, která část zpracování brala nejvíce času, a následně provést potřebné úpravy.

Po následném přepsání do asynchronní detekce byla dosažena snímková frekvence kolem 28 fps.



Obrázek 6 Segmenty měření

9 Implementace herní scény v Unity

Můj hlavní cíl v tomto projektu nebyl vytvořit jednu konkrétní hru, ale vyvinout základní systém pro sledování rukou pomocí MediaPipe a RealSense v Unity, který by bylo možné dále rozšiřovat a využít k tvorbě různých aplikací.

I když byla implementace herní scény součástí osnovy, hlavní výzvou nebylo vytvoření samotné technologie, ale její správná integrace tak, aby umožňovala precizní sledování pohybu rukou ve 3D prostoru a interakci s herním prostředím.

9.1 Koncept hry ve stylu BeatSaber

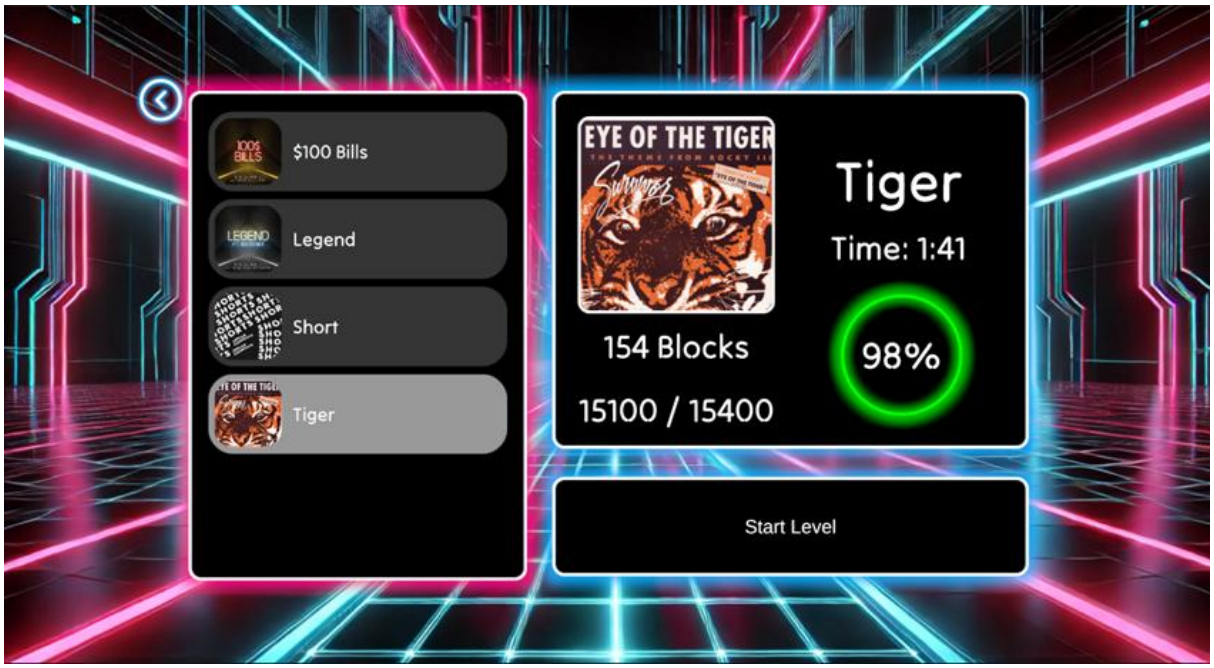
Na začátku projektu nebyl konkrétní koncept hry jasně definován. Existovalo několik různých nápadů, jak technologii využít, například aplikace na malování, kde by se prstem kreslilo do prostoru, nebo simulace zachytávání objektů v prostoru, kde by hráč mohl interagovat s virtuálními prvky pouhým gestem.

Nakonec bylo rozhodnuto pro koncept inspirovaný hrou Beat Saber, protože tento styl hry efektivně využívá přesné sledování pohybů rukou, je dynamický a dobře se hodí pro technologii, která byla implementována. Tato hra ale není přesnou kopií Beat Saberu – funguje na podobném principu, pouze s určitými rozdíly. Z prostoru přilétají bloky, které je nutné zničit rukama, přičemž hra vyhodnocuje úspěšnost zásahů. Díky tomu se vytváří interaktivní zážitek, ve kterém hráč fyzicky interaguje s virtuálním světem pouze pohybem rukou, bez potřeby jakýchkoli ovladačů.

9.2 Herní menu

Herní menu slouží jako hlavní navigační prvek, který umožňuje hráči vybírat jednotlivé úrovně a zobrazit relevantní informace o jejich průběhu. Každý level má přiřazené maximální dosažené skóre, které se zobrazí hráči před spuštěním hry. Součástí výběru úrovně je také přiřazená hudba, která se spustí společně s daným levelem.

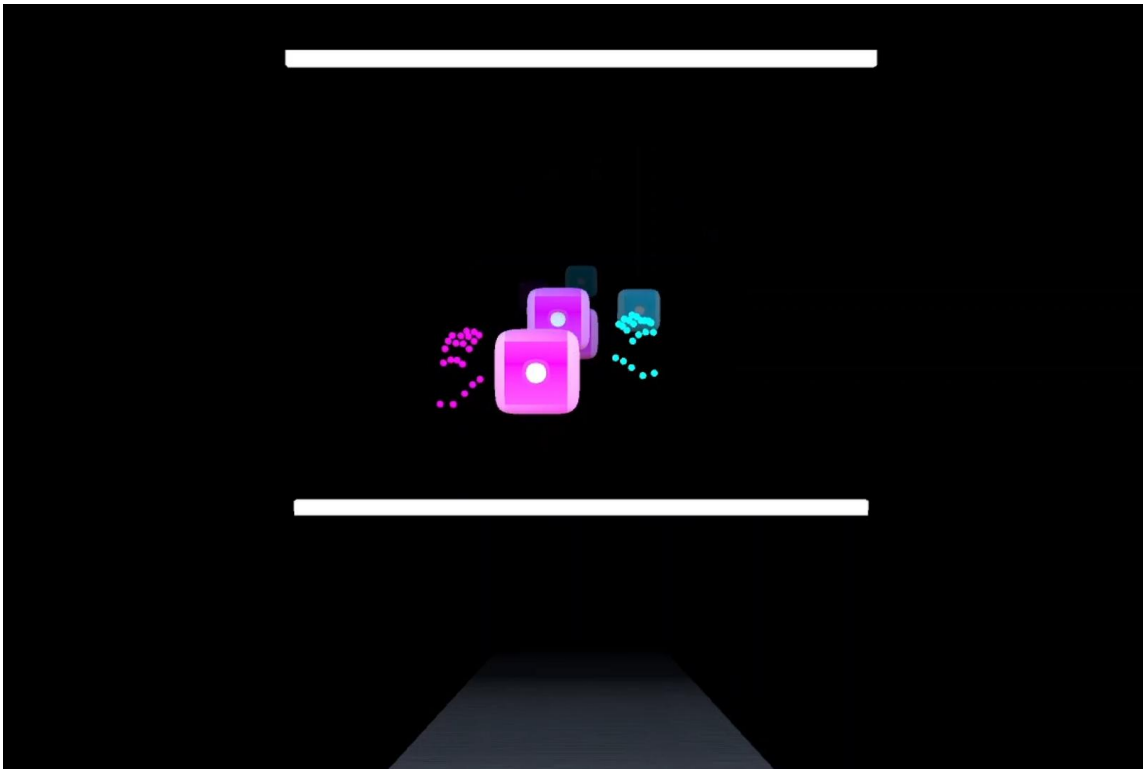
Menu je navrženo tak, aby bylo jednoduché a intuitivní, čímž hráči poskytuje přehlednou možnost volby a motivaci ke zlepšování výsledků. Díky tomuto systému může hráč opakovaně procházet jednotlivé úrovně a snažit se dosáhnout lepšího skóre, což přispívá ke zvýšení znovuhratelnosti hry.



Obrázek 7 Výběr levelu v menu

9.3 Hlavní herní scéna

Hlavní herní scéna představuje dynamické prostředí, ve kterém hráč interaguje s objekty v reálném čase. Klíčovým prvkem hry je generování letících bloků, které hráč musí zničit pohybem rukou. Tento proces je synchronizován s hudbou, což zajišťuje plynulý a rytmický zážitek. Hráč se tak pohybuje v souladu s hudbou a jeho úkolem je správně reagovat na přilétající bloky.



Obrázek 8 Herní scéna

9.3.1 Generování letících „bloků“

Pro generování bloků byl vytvořen Python skript, který analyzuje hudební skladbu a extrahuje z ní beaty. Získané informace se ukládají do JSON souboru, jenž definuje časování a vlastnosti jednotlivých bloků v úrovni. Díky tomuto postupu se bloky objevují ve hře synchronně s hudebními beaty.

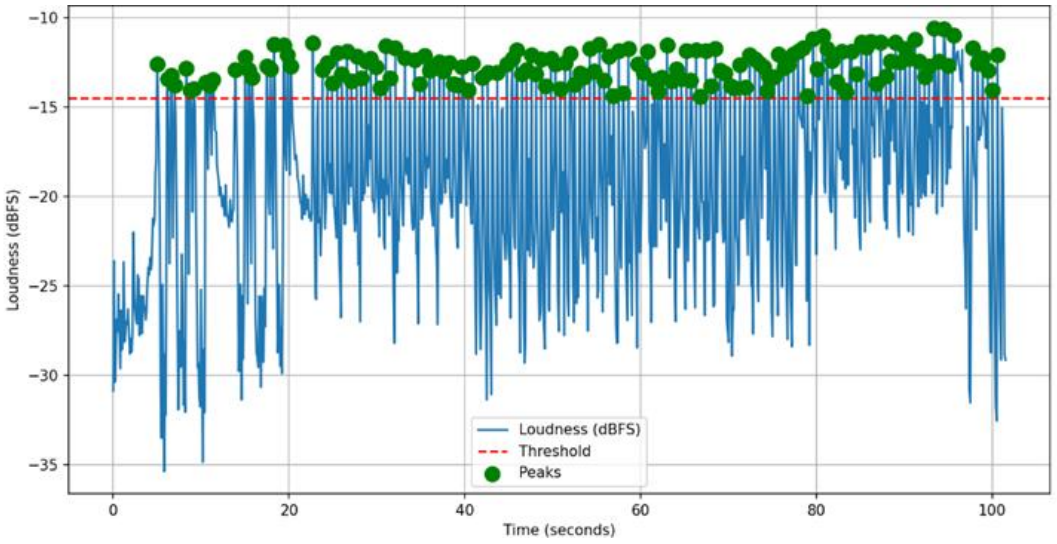
V Unity je pak během hry JSON soubor načten a pokud aktuální čas ve hře odpovídá času bloku v souboru, blok se v daný moment objeví na herní scéně. Tím se zajišťuje, že všechny bloky přilétají ve správný okamžik a v souladu s rytmem hudby.

9.3.1.1 Detekce beatů v audiu

Skript využívá knihovnu pydub ke zpracování audia a rozdělení zvukového signálu do krátkých segmentů. Pro každý segment se vyhodnocuje hlasitost (dBFS) a pokud překročí nastavenou prahovou hodnotu, zaznamená se časová značka odpovídající vrcholu dané oblasti. Níže je ukázka klíčové části kódu, kde se analyzují jednotlivé audio segmenty:

```
for i, chunk in enumerate(self.chunks): #Iterace přes chunky
    if chunk.dBFS > self.threshold_dbfs:
        if region_start is None:
            region_start = i
            region_max_dbfs = chunk.dBFS
            region_max_index = i
        else:
            if chunk.dBFS > region_max_dbfs:
                region_max_dbfs = chunk.dBFS
                region_max_index = i
    else:
        if region_start is not None:
            timestamps.append(region_max_index *
self.chunk_size_ms / 1000.0)
            self.peak_indices.append(region_max_index)
            region_start = None
            region_max_dbfs = -float('inf')
            region_max_index = None
```

Tento úsek kódu zajišťuje, že se do výsledného seznamu zapíše pouze jeden časový okamžik (peak) pro každou oblast, kde hlasitost přesáhne prahovou hodnotu.



Obrázek 9 Graf hlasitosti audia

9.3.1.2 Generace JSON souboru levelu

Na základě detekovaných časových značek se následně vytvoří JSON soubor, který definuje vlastnosti úrovně a jednotlivých bloků. Každý blok obsahuje informace jako je čas, pozice, typ bloku, informace o tom, zda je určen pro pravou nebo levou ruku, a bodová hodnota. Níže je uveden výňatek z kódu, který generuje objekt úrovně a ukládá jej do JSON souboru.

```
block = {
    "id": i + 1,
    "time": time_value,
    "position": {
        "x": round(x_pos, 3),
        "y": round(y_pos, 3)
    },
    "type": "standard",
    "isRight": is_right,
    "points": block_points
}
blocks.append(block)
```

9.3.2 Využití kolizí pro detekci zásahu rukou

Aby bylo možné správně detekovat, zda hráč zasáhl blok rukou, bylo nutné využít systém kolizí v Unity (11). Každý blok je objekt s RigidBody, což umožňuje jeho detekci v prostoru a správné chování v rámci fyzikální simulace. Na druhé straně jsou na pozicích rukou umístěny collidery, které odpovídají landmarkům ze sledování rukou pomocí MediaPipe a RealSense.

Každý blok má přiřazenu informaci, zda se má trefit pravou nebo levou rukou. Pokud hráč zasáhne blok špatnou rukou, dostane negativní skóre, pokud správnou, skóre se mu zvýší.

V Unity se kolize detekuje metodou OnTriggerEnter(Collider other), která ověřuje, zda se blok srazil s některým z ručních colliderů (JointL pro levou ruku a JointR pro pravou).

```
private string LeftCollider = "JointL";
private string RightCollider = "JointR";

private void OnTriggerEnter(Collider other)
{
    int scoreDelta = 0;
    if (other.CompareTag(RightCollider))
        scoreDelta = isRight ? score : -score;
    else if (other.CompareTag(LeftCollider))
        scoreDelta = isRight ? -score : score;
    LevelLoader.instance.addScore(scoreDelta);
    DestroyBlock();
}
```

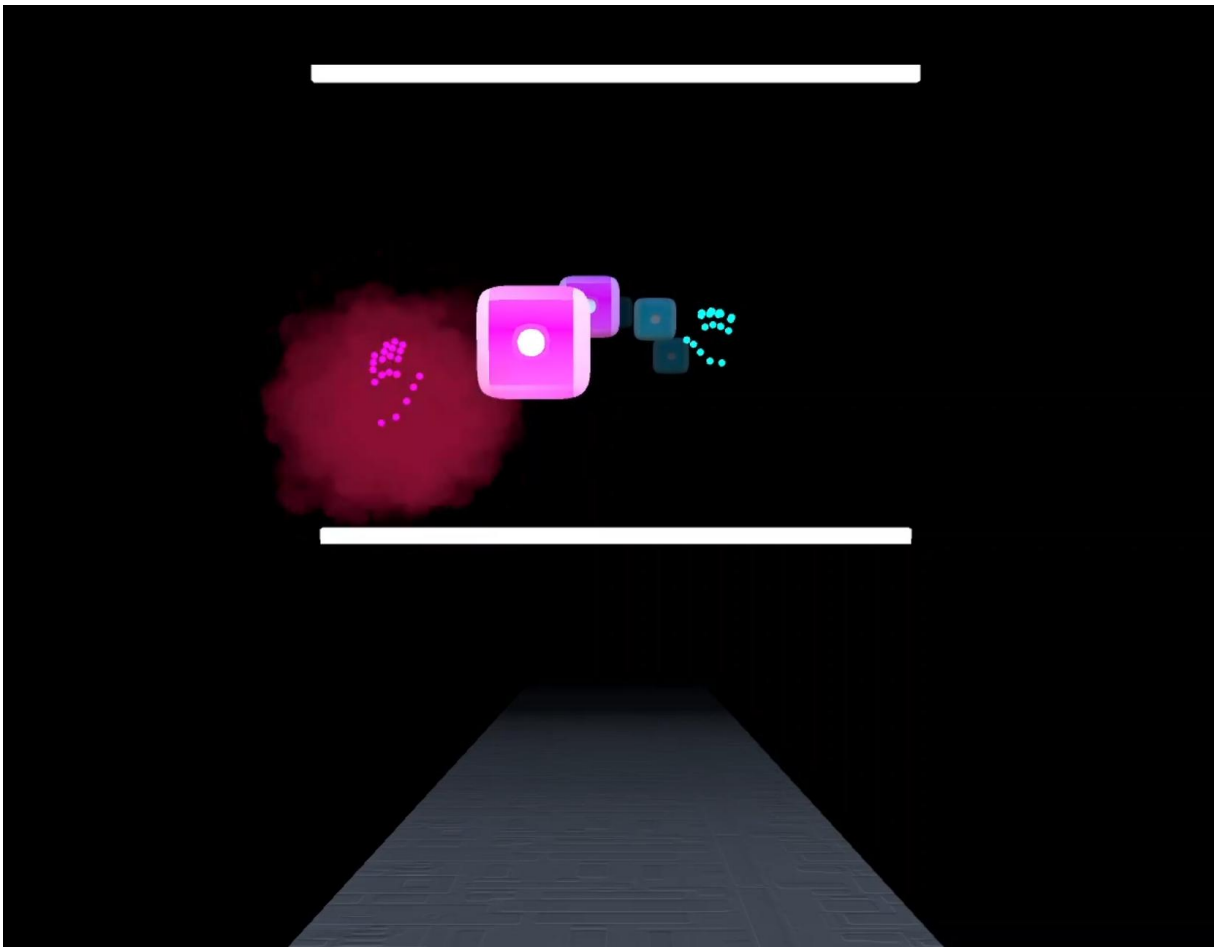
9.3.3 Destrukce bloků a odezva

Samotné zmizení bloku po zásahu by nebylo dostatečně atraktivní, a proto je destrukce bloků doplněna o vizuální a zvukové efekty, které zlepšují hráčský zážitek.

Při zničení bloku dojde k zobrazení částicového efektu (12), který simuluje explodující blok. Tento efekt zajišťuje, že vizuální odezva na zásah je dynamická a přirozená. Současně se přehraje krátký zvukový efekt, který signalizuje úspěšný zásah a poskytuje hráči okamžitou zvukovou zpětnou vazbu. Tyto prvky dohromady vytvářejí plynulý a interaktivní dojem ze hry.

```
private void DestroyBlock()
{
    var effectvariation = isRight ? destroyEffectR :
destroyEffectL;
    var effect = Instantiate(effectvariation,
transform.position, Quaternion.identity);
    Destroy(effect,
effect.GetComponent<ParticleSystem>().main.duration);
    LevelLoader.instance.playSound(destroySound);
    Destroy(gameObject);
}
```

Tento systém přidává zpětnou vazbu hráči, což zlepšuje celkový herní zážitek a činí hru zábavnější.



Obrázek 10 Zobrazení efektu rozbití bloku levou rukou

10 Možnosti rozšíření a budoucí vývoj

Výkon a plynulost aplikace mohou být dále optimalizovány. Jednou z možností je zrychlení zpracování snímků, například optimalizací algoritmů a využitím efektivnějších metod pro načítání dat z kamer. Snížení latence mezi zachycením a zobrazením dat by zajistilo, že reakce aplikace bude téměř okamžitá, což je klíčové pro interaktivní hry v reálném čase.

Detekční algoritmy by mohly být vylepšeny tak, aby byla detekce rukou ještě přesnější a robustnější. Modernizace těchto algoritmů by umožnila lépe reagovat na různé světelné podmínky a proměnlivé pozice rukou, čímž by se zvýšila spolehlivost a přesnost sledování, a to i v náročnějších situacích.

Herní scéna může být rozšířena o více úrovní, přičemž každá úroveň by nabízela odlišné herní mechaniky a výzvy. Tím by byla zajištěna větší pestrost a opakovatelnost hry, což by přispělo k delší době hraní a lepšímu zážitku pro hráče.

Závěr

Během práce jsem se setkal s řadou problémů, které mě přinutily strávit hodiny studiem dokumentace a hledáním řešení. Každá překážka byla zároveň příležitostí se něco nového naučit – ať už šlo o správnou integraci senzorů, nebo o to, jak lépe zpracovávat data. Měl jsem možnost provádět vlastní výzkum, což mi pomohlo lépe pochopit, jak celý systém funguje, a posunulo to moje znalosti na vyšší úroveň.

Celkově si myslím, že výstup mé práce splňuje všechna stanovená kritéria. Dokázal jsem vytvořit funkční a spolehlivý systém, který spojuje technologie jako Intel RealSense a MediaPipe, a umožňuje přesné sledování pohybu rukou. Projekt pro mě znamenal skvělou zkušenost, protože jsem se naučil hodně nového, což mi otevřelo dveře do světa interaktivních technologií.

Seznam zkratek a odborných výrazů

MediaPipe
Knihovna pro zpracování obrazu a detekci klíčových bodů ruky pomocí strojového učení.

RealSense
Technologie od Intelu pro hloubkové snímání 3D prostoru.

Unity
Herní engine používaný pro vývoj interaktivních aplikací a her.

Android
Operační systém pro mobilní zařízení, na kterém běží výsledná aplikace.

UPBGE
Fork Blenderu s herním enginem – upravená verze Blender Game Engine.

.NET
Platforma od Microsoftu pro vývoj softwaru, využívaná pro tvorbu knihoven.

C#
Programovací jazyk (C Sharp), hlavní jazyk používaný v Unity.

RGB
Barevný model založený na kombinaci červené, zelené a modré.

FPS
Frames Per Second – počet snímků zobrazených za sekundu.

dBFS
Decibels relative to Full Scale – měření hlasitosti digitálního signálu.

NPU
Neural Processing Unit – čip určený pro zpracování úloh spojených se strojovým učením.

AAR
Android Archive – formát pro distribuování a integraci knihoven do Android aplikací.

DLL
Dynamic Link Library – knihovna dynamicky linkovaného kódu, využívaná pro volání nativních funkcí.

Seznam obrázků

Obrázek 1 Konstrukce krabičky 4
Obrázek 2 Testování MediaPipe a Intel RealSense v Pythonu 6
Obrázek 3 MediaPipe a RealSense v Unity 6
Obrázek 4 Časová osa projektu 6
Obrázek 5 21 klíčových bodů ruky (10) 13
Obrázek 6 Segmenty měření 15
Obrázek 7 Výběr levelu v menu 17
Obrázek 8 Herní scéna 17
Obrázek 9 Graf hlasitosti audia 19
Obrázek 10 Zobrazení efektu rozbití bloku levou rukou 21

Použité zdroje

1. **Radxa.** Radxa ROCK 5B+. *radxa.* [Online] radxa. [Citace: 10. březen 2025.] https://radxa.com/products/rock5/5bp#techspec.
2. —. Summary of resource downloads. *radxa Documentation Center.* [Online] [Citace: 10. březen 2025.] https://docs.radxa.com/en/rock5/rock5b/download.
3. **Intel Corporation.** Android for Rooted Devices. *Intel Realsense.* [Online] 1. leden 2021. [Citace: 10. březen 2025.] https://dev.intelrealsense.com/docs/android-for-rooted-devices.
4. —. Depth Camera D415. *intel REALSENSE.* [Online] [Citace: 10. březen 2025.] https://www.intelrealsense.com/depth-camera-d415/.
5. —. Build RealSense SDK for Android OS. *GitHub.* [Online] 19. únor 2019. [Citace: 9. březen 2025.] https://github.com/IntelRealSense/librealsense/blob/master/wrappers/android/readme.md.
6. —. Build an Android application for Intel RealSense SDK. *intel realsense.* [Online] 1. leden 2023. [Citace: 10. březen 2025.] https://dev.intelrealsense.com/docs/build-an-android-application-for-intel-realsense-sdk.
7. **Unity Technologies.** Call Java and Kotlin plug-in code from C# scripts. *Unity Documentation.* [Online] 10. březen 2025. [Citace: 11. březen 2025.] https://docs.unity3d.com/6000.0/Documentation/Manual/android-plugins-java-code-from-c-sharp.html.
8. **Nishida, Junrou.** Build Instructions. *GitHub.* [Online] 17. únor 2025. [Citace: 3. březen 2025.] https://github.com/homuler/MediaPipeUnityPlugin/blob/master/docs/Build.md.
9. **Microsoft.** DllImportAttribute Class. *Microsoft Learn.* [Online] [Citace: 10. březen 2025.] https://learn.microsoft.com/en-us/dotnet/api/system.runtime.interopservices.dllimportattribute.
10. **Google AI for Developers.** Hand landmarks detection guide. *Google AI for Developers.* [Online] 13. leden 2025. [Citace: 10. březen 2025.] https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker.
11. **Unity Technologies.** Collider.OnTriggerEnter(Collider). *Unity Documentation.* [Online] 10. březen 2025. [Citace: 11. březen 2025.] https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Collider.OnTriggerEnter.html.
12. —. Particle systems. *Unity Documentation.* [Online] 10. březen 2025. [Citace: 11. březen 2025.]

https://docs.unity3d.com/6000.0/Documentation/Manual/ParticleSystems.html
.

13. **Stehlík, Michal.** *Návod k maturitním pracím 2020.* Liberec : Albatros, 2020.

A. Seznam přiložených souborů

Odkaz na příslušný repozitář MP:

- https://github.com/pslib-cz/MP2024-25_Adamec-Matyas_MediaPipe-pro-Unity

Ve zmíněném repozitáři se nacházejí následující soubory a složky:

- **MP2025-Adamec-Matyáš -P4A-MediaPipe-pro-Unity.docx** – editovatelná verze dokumentace maturitní práce
- **MP2025-Adamec-Matyáš -P4A-MediaPipe-pro-Unity.pdf** – tisknutelná verze dokumentace maturitní práce
- **Aplikace** – zdrojové kódy