ÚSTAV AUTOMATIZACE A INFORMATIKY

Simulace dynamických systémů

Jméno: Jakub Michalica		ID studenta: 208897
Datum zadání:	Datum odevzdání: 15.01.2024	Studijní skupina: 5pAIŘ/1
Název úlohy: **Výstřel z palné zbraně**		

ZADÁNÍ ÚLOHY

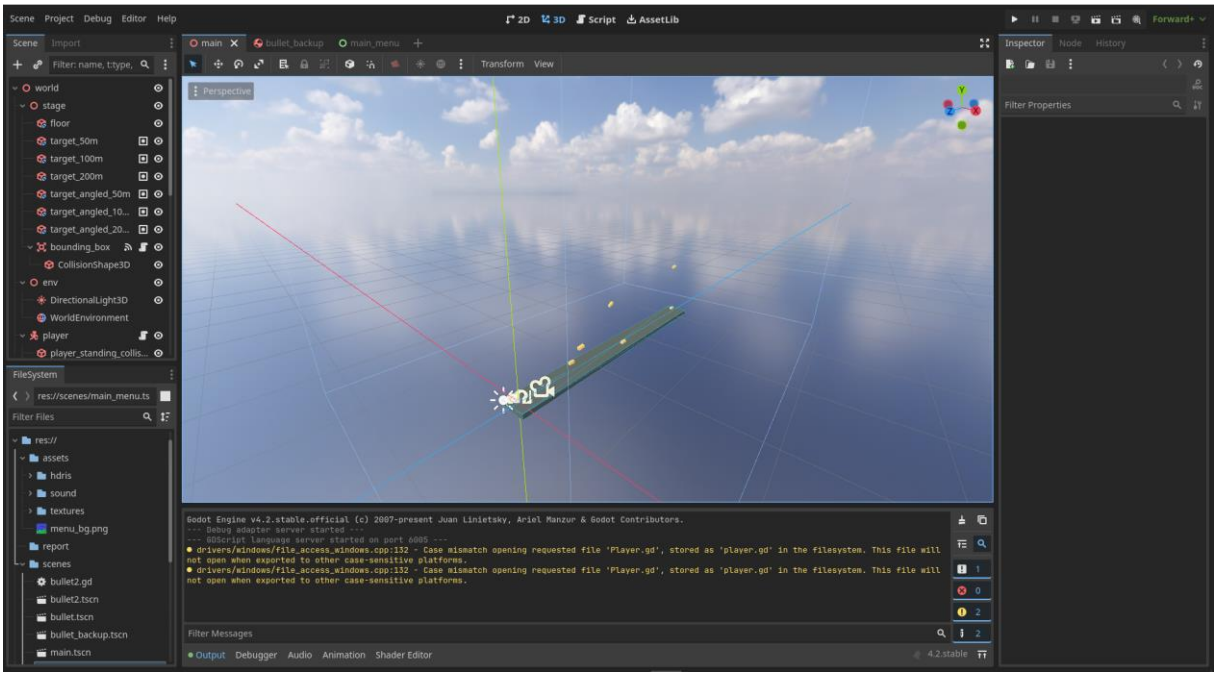
Simulujte spojitý dynamický systém, vámi zvolenou metodou a implementací. Jako zadání jsem si vybral „výstřel z palné zbraně“, který jsem se rozhodl zpracovat jako simulaci letu projektilu ve 3D pomocí herního enginu Godot.

GODOT GAME ENGINE

Godot je bezplatný, open-source game engine vydaný v roce 2014. Jedná se o multiplatformní nástroj dostupný na Windows, MacOS, Linux, a dokonce i ve webovém prostředí a mimo desktopové platformy podporuje export pro Android, HTML5 a WebAssembly.

Vývoj v Godot se zakládá na tvorbě „scén“, které mohou reprezentovat například menu nebo jednotlivé úrovně atd. Samotné scény se poté skládají z jednotlivých „uzlů“ (Nodes), které definují buďto samotnou scénu nebo jednotlivé prvky uvnitř scény. Godot poskytuje tři základní uzly a to 2D, 3D a UI. 2D a 3D uzly definují zda bude daná scéna ve 2D nebo 3D prostředí a UI uzel umožňuje tvorbu uživatelských rozhraní jako například menu, nastavení, textová pole atd.

Ke každému uzlu lze připojit skript, který bude následně definovat nebo modifikovat jeho chování. Pro psaní skriptů jsou je k dispozici interní skriptovací jazyk GDScript, jedná se o dynamicky typovaný objektově orientovaný jazyk. Dále jsou oficiálně k dispozici C# nebo C++ anebo komunitní implementace pro Python, Rust, Swift a další.



Obrázek 1 - Okno Godot editoru pro 3D prostředí

VÝPOČET TRAJEKTORIE

Původně jsem měl v plánu simulovat let kulky podle balistických manuálů a implementovat více ráží a komplexnější povětrnostní podmínky. Bohužel vzhledem k mé nezkušenosti, co se práce s jakýmkoli herním enginem týče, mi dalo celkem dost práce zjistit, jak věci fungují a hlavně jak správně vyřešit výpočet trajektorie a rychlostí tak aby výsledek nebyl aplikován pouze pro lokální orientační systém střely, což sice způsobovalo vcelku vtipné chování, ale nebylo to to co jsem si přál.

Ve výsledné implementaci používám pro výpočet dva hlavní vzorce:

$$v_x = v_{ix} - a_{wind} \cdot \Delta \; [ms^{-1}]$$

$$v_y = v_{iy} - g \cdot \Delta \; [ms^{-1}]$$

$$v_z = v_{iz} \; [ms^{-1}]$$

Kde $v_{x,y,z}$ je rychlost v dané ose, v_i je počáteční rychlost projektilu, g gravitační zrychlení, a_{wind} zrychlení způsobené působením větru a Δ je krok času, neboli proměnná „delta“ určující časový krok v rámci Godot.

Výpočet a_{wind}

Pro výpočet zrychlení od větru jsem vycházel z:

$$a_{wind} = \frac{F_w}{m} \; [ms^{-2}][1]$$

Kde m je hmotnost kulky a F_w síla vzniklá působením větru na kulku. Pro hmotnost kulky jsem uvažoval že je vyrobena z olova (ρ = 11340 kg/m^3 [2]) a jako objem jsem považoval objem 3D modelu použitého pro její reprezentaci.

$$m = \rho \cdot V\ [kg]$$

Výpočet síly větru:

$$F_w = A \cdot P \cdot C_d [N][3]$$

Kde A je plocha řezu kulkou v rovině YZ, C_d je koeficient odporu vzduchu uvažovaný jako C_d = 0.47 [4] (což je hodnota pro kouli) a P je tlak způsobený rychlostí vzduchu v_{wind}.

$$P = \frac{1}{2} \cdot \rho_w \cdot v_{wind}^2\ [Pa][5]$$

Hustotu vzduchu jsem uvažoval jako ρ_w = 1.293 kg/m^3 [6].

Tato implementace působení větru není ani zdaleka dokonalá, ale funguje a její působení lze dobře pozorovat například při větru o rychlosti 20 000 m/s (což je cca 58násobek rychlosti zvuku).

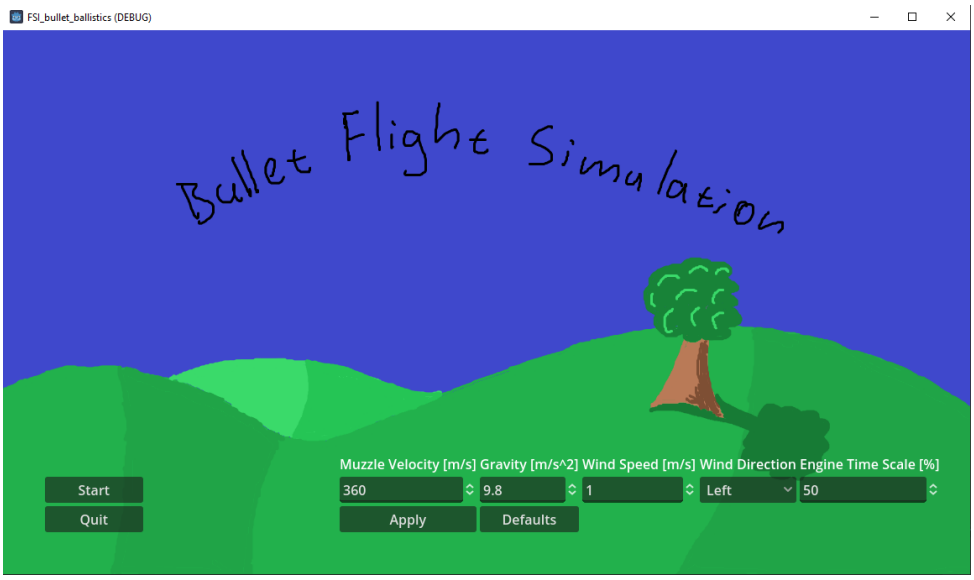
POPIS VÝSLEDNÉ IMPLEMENTACE

Má výsledná implementace se skládá ze tří scén: Hlavní menu, hlavní scéna a samotná kulka.

Hlavní menu

Toto menu slouží k úpravě nebo nastavení parametrů při prvním spuštění nebo se do něj lze vrátit pomocí klávesy ESC. Po nastavení požadovaných parametrů je potřeba zmáčknout tlačítko „Apply“ aby se změny propsali do globálního skriptu který slouží k ukládání proměnných používaný ve vícero scénách. Tlačítko „Defaults“ naopak slouží pro navrácení parametrů na původní hodnoty. Do hlavní scény se poté přepne pomocí tlačítka „Start“.

Parametr „Engine time scale“ slouží k nastavení rychlosti jakou poběží samotny engine. Jedná se o procentuální hodnotu kde 100 % je normální rychlost a vše pod 100% znamená že celý software poběží pomaleji. Tato proměnná je zde pro případ kdy by docházelo k problémům s kolizemi v případě příliš velké úsťové rychlosti kulky což může nastat i potom co je frekvence výpočtů fyzikálních procesů nastavena na 480Hz z původních 60Hz.



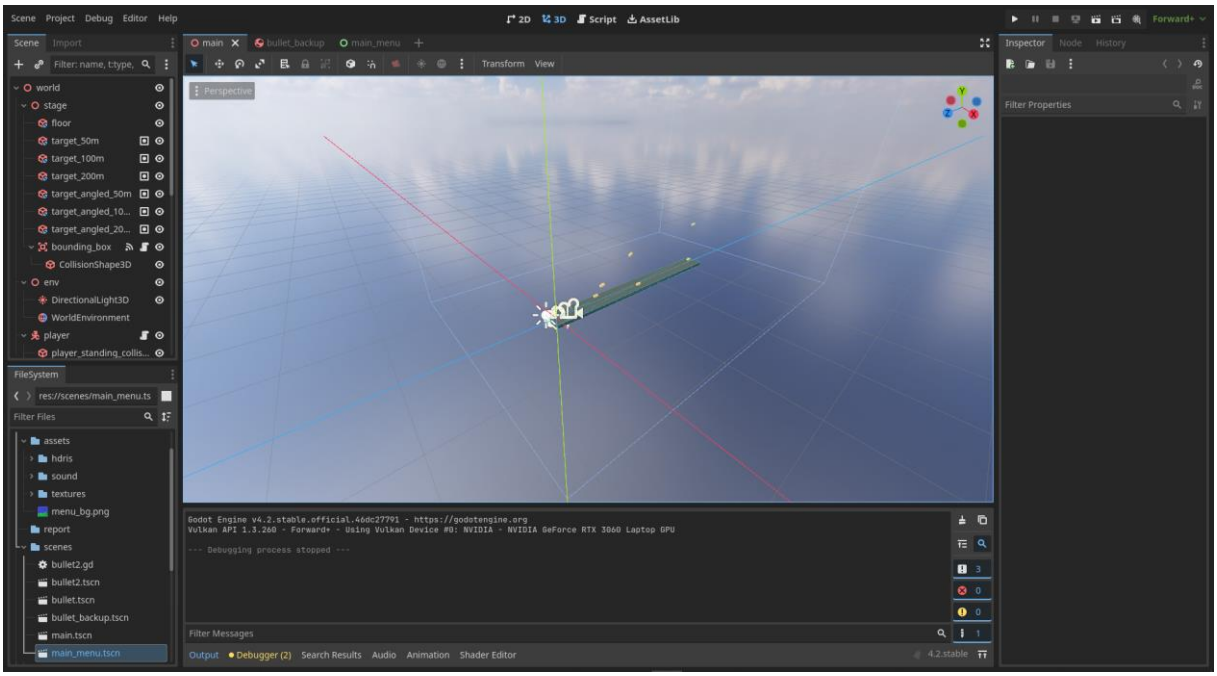
Obrázek 2 - Hlavní menu

Hlavní scéna

V této scéně je sestavena jednoduchá střelnice s terči ve vzdálenostech 50, 100 a 200 metrů, do kterých se může uživatel zkoušet trefit. Vždy je možné vystřelit pouze jednou a následně počkat kam kulka dopadne. Uprostřed horní hrany okna lze vidět, zda poslední kulka zasáhla nebo nezasáhla terč, v případě zasažení terče se navíc ozve zvukový efekt. Dále je možné po pravém okraji obrazovky pozorovat rychlosti kulky ve všech osách, potvrzení, zda instance kulky stále existuje a zda má kamera následovat vystřelenou kulku. Na levé straně jsou informace o úsťové rychlosti, rychlosti větru a jeho směru, gravitačním zrychlení (kladná hodnota znamená že gravitace působí směrem dolů), rychlosti enginu a frekvenci fyzikálních výpočtů.

Mimo míření myší a výstřelu pomocí levého tlačítka myši má uživatel k dispozici i množství klávesových zkratek, které umožňují změnu parametrů simulace bez nutnosti návratu do menu a několik dalších funkcí. Všechny klávesové zkratky jsou dostupné v dokumentu README.md přiloženém v projektu nebo přímo na GitHub stránce projektu. Zde uvedu pouze několik zkratek, které jsou dle mého názoru nejvíce užitečné.

Klávesa	Funkce
LMB	Výstřel
RMB	Změna kamery
E	Vypnutí/zapnutí následování vystřelené kulky
Esc	Návrat do menu



Obrázek 3 - Hlavní scéna v editoru

Na obrázku 3 jde vidět, že střelnici obklopuje neviditelný kvádr. Ten detekuje, jestli kulka opustila jeho objem, pokud ano okamžitě ji smaže. Tímto zabraňuje tomu aby náhodou kulka neodletěla mimo střelnici díky tomu, že už nikdy nic nezasáhne, tak by nikdy nezmizela a uživatel by nemohl znovu vystřelit bez restartu celého programu.

Obrázek 4 - Spuštěná hlavní scéna

Kulka

Tato scéna slouží k definici samotné kulky a zároveň výpočtu jejího letu. Pokaždé když má být kulka vystřelena, vytvoří se instance této scény a umístí se na konec hlavně zbraně jako dítě tohoto uzlu. Následně se hlavní scéna připojí na signál, který je vyslán v případě zásahu terče a poté zavolá funkci kulky která způsobí její vystřelení. Jako poslední krok hlavní scéna spustí audio přehrávač uvnitř kulky k přehrání zvuku výstřelu.

```
84      # Fire bullet
85      if Input.is_action_just_pressed("fire"):
86          # If bullet does not exist (limits number of bullets to one)
87          if !is_instance_valid(b):
88              # Set "target hit" to false
89              update_target_label(false)
90              # Create instance of a bullet
91              b = bullet.instantiate()
92              # Place the bullet at the muzzle node
93              muzzle.add_child(b)
94              # Switch camera to camera on the bullet
95              if follow_cam:
96                  b.activate_cam()
97              # Connect signal that detects if target was hit
98              b.connect("target_hit", update_target_label_signal)
99              # fire the bullet
100             b.fire(player_head.global_rotation)
101             # Reset bullet's audio player volume
102             b.bullet_player.volume_db = 0.0
103             # Set firing sound
104             b.bullet_player.stream = shot_sound
105             # Play firing sound
106             b.bullet_player.play()
107             #print("hit")
```

Obrázek 5 - Kód pro výstřel uvnitř hlavní scény

```
43  func _ready():
44      # When created as a child it instantly becomes it's own object so it can't
45      # be controlled by the changes in parrent
46      set_as_top_level(true)
47
```

```
93  func fire(muzzle_rotation):
94      # apply initial firing impulse to the bullet
95      # Rotate the bullet so it's rotated the same way as the gun muzzle
96      rotation = muzzle_rotation
97      # Apply initial muzzle velocity
98      new_velocity = -transform.basis.z * Global.initial_velocity
99      was_shot = true
100
```

Obrázek 6 - Kód pro výstřel uvnitř kulky

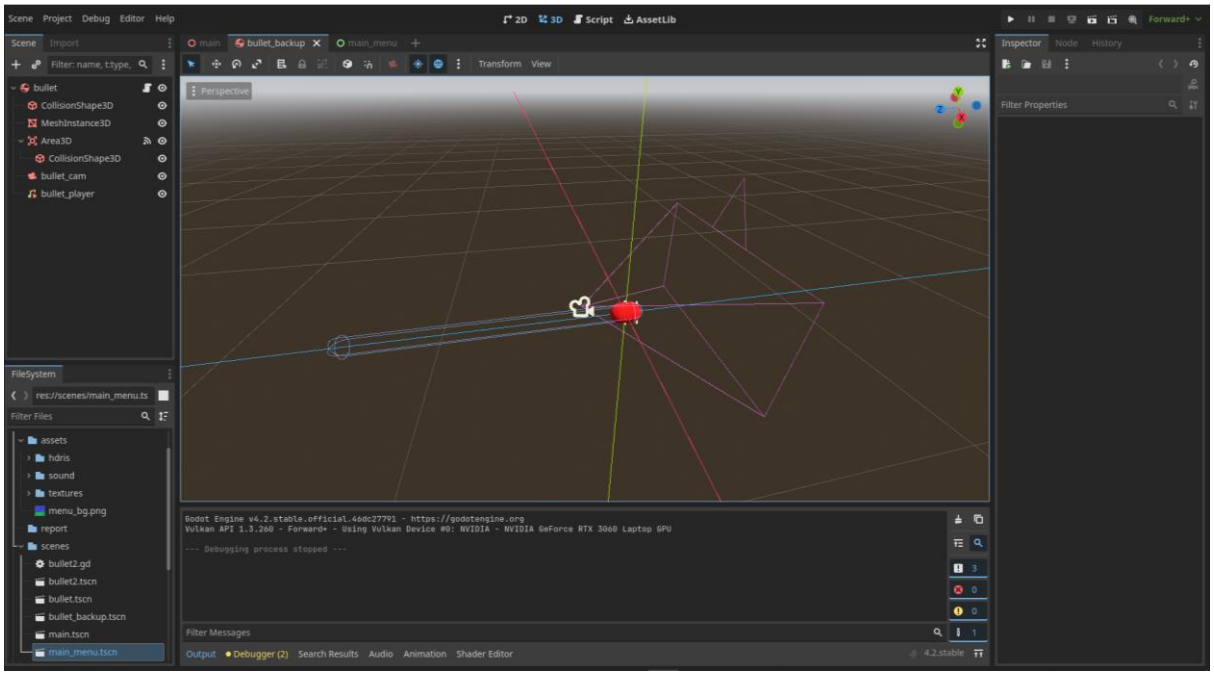
Uvnitř kulky se po vytvoření její instance ve funkci „_ready()“ se kulka odpojí od jakéhokoli rodiče a to z důvodu aby její pohyb nemohl být ovlivněn jiným uzlem. Následně ve funkci „fire()“ se kulka natočí stejným směrem jako byla natočena hlaveň zbraně a její rychlost se nastaví jako úsťová rychlost ve směru -Z což je směr reprezentující dopředu ve scéně kulky.

Samotný výpočet letu kulky se poté počítá ve funkci „_physics_process()“. Tato funkce slouží k výpočtům ovlivňujícím fyzikální chování a lze zde naprogramovat vlastní chování bez toho aniž by to ovlivnilo interní logiku pro detekci kolizí. Můj kód zde reprezentuje rovnice uvedené v předchozí kapitole.

```gdscript
# Called every frame. 'delta' is the elapsed time since the previous frame.
func _physics_process(delta):
	new_velocity += g * delta

	## Calculation of the velocity change due to wind
	# wind preassure
	preassure = 0.5 * rho_air * (Global.wind_speed**2)
	# Wind force
	force = area * preassure * drag
	# Wind acceleration
	a = force / m
	# Final wind speed
	v_f = a * delta

	if Global.wind_speed < 0:
		v_f *= -1.0

	w = Vector3.LEFT * v_f
	#w = Vector3.LEFT * Global.wind_speed
	new_velocity += w * delta
	look_at(transform.origin + new_velocity.normalized(), Vector3.UP)
	transform.origin += (new_velocity * delta)
```

Obrázek 7 - Výpočet letu kulky

Jako poslední chci zmínit samotnou 3D scénu kulky, kde lze na obrázku 8 vidět její složení. Hlavními členy jsou zde samotný červený model kulky, kamera, která je používána pro následování kulky a její kolizní tvar. Kolizní tvar nekopíruje přesně model, ale je značně protažený dozadu. To je ze snahy zlepši detekci kolizí pro vyšší rychlosti, kde může nastat, že v jednom snímku je kulka před terčem a v druhém za. Pokud je kolize kulky i za ní je větší šance, že alespoň nějaká část bude uvnitř cíle a spustí detekci kolizí



Obrázek 8 - Scéna kulky uvnitř editoru

ZÁVĚR

Celkově jsem se svým zpracováním projektu spokojený, i přes to, že se nejedná o simulaci. Jsem za možnost si díky tomuto projektu rozšířit obzory a zkušenosti a vyzkoušet práci v Godot enginu, který má dle mého názoru velkou budoucnost díky svým relativně nízkým nárokům na hardware, tak i uživatele a tím, že se jedná o open-source multiplatformní software.