**Documentația proiectului aMAZEing**

Membri:

* Dincă Vlad-Andrei
* Gavrilă Alexandru-Ionuț
* Istudor Georgiana-Cristina

Proiectul este un joc endless ce se desfășoară într-un labirint. Cu cât termini mai multe labirinturi fără să fii omorât esti mai aproape de un high-score-ul personal. Secretul unui rank înalt este să termini labirintul fără să fii rănit de către inamici, scorul se resetează atunci când health-ul ajunge la 0. Pentru a crește scorul trebuie sa iți termini misiunea și să omori un număr de adversari. Când misiunea este incheiată, se deschide un portal catre labirintul următor, unul complet nou.

Generarea labirintului se face folosind algoritmul Growing Tree.

1. Fie activeCells o listă de celule, inițial goală. Adăugăm o celulă random la activeCells.
2. Alegem o celulă din activeCells, selectez întotdeauna ultimul indice, ceea ce face ca algoritmul să se scufunde pe căi înguste care circulă pe tot labirintul, și realizăm un pasaj la orice vecin nevăzut al acelei celule, adăugându-l și pe vecinul lui activeCells. Pentru a umple complet labirintul, ar trebui să eliminăm o celulă din lista activă numai atunci când toate marginile sale au fost inițializate. Acesta este primul lucru pe care ar trebui să îl verificăm, deoarece o celulă din lista activă va fi inițializată complet atunci când toți vecinii au fost vizitați. Și pentru a preveni plasarea incorectă a pereților, ar trebui să alegem doar o direcție aleatorie care nu este încă inițializată pentru celula actuală.
3. Repetați numărul 2 până când activeCells este gol.

Ușile sunt un alt element interesant adăugat in labirint. Vrem să creăm uși în loc de pasaje, dar nu chiar atât de des, deoarece altfel labirintul va fi inundat cu uși. Deci adăugăm o opțiune de configurare a probabilității ușii și o utilizăm pentru a decide dacă plasăm o ușă sau un pasaj.

Crearea primei camere se face la pasul 1 din generarea labirintului. Pentru a pune toate celelalte celule într-o cameră, verificam dacă o ușă a fost plasată. Dacă da, cealaltă celulă este prima dintr-o încăpere nouă. Dacă nu, aparține aceleiași camere ca celula anterioară.

Mecanicile de miscare sunt simpliste, jucatorul neavand posibiltatea sa alerge sau sa sara, deoarece labirintul nu este unul mare si nici nu avem obstacole sau verticalitate. Pentru a realiza miscarea s-a folosit un Character Controller.

Au fost adaugate sunete[14] pentru diferite actiuni in joc: pasi jucator, arma jucator, arma inamic, teleporter. Au fost adaugate folosind un obiect de tip Audio Source si sunt redate atunci cand se indeplinesc anumite conditii(jucatorul se misca si are o viteza suficient de mare, se trage cu arma sau apare teleporterul pe harta).

Pentru un plus de finisaj au fost adaugate diferite texturi de caramida[4] pentru pentru pereti si podea, textura de lemn pentru usi[2] si un skybox[1].

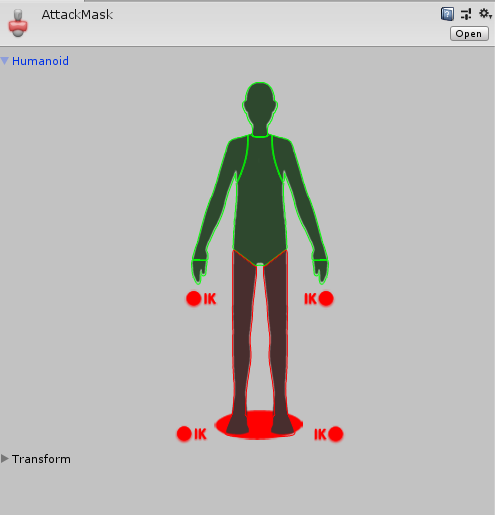
Teleporterul [3] este un model de pe asset store in jurul caruia a fost adaugat un trigger pentru a detecta momentul cand player-ul doreste sa treaca la urmatorul nivel. Acesta se genereaza intr-o celula ce nu contine pereti atunci cand jucatorul a omorat numarul minim de inamici din labirint.

Atunci cand jucatorul atinge un nou high-score pe ecranul “Game Over’ va aparea un mesaj care as anunte acest lucru. Jucatorul va avea optiunea sa salveze acest scor prin inserarea unui nume si apasarea butonului “Save”.

Modelul 3D al inamicilor este Space Robot Kyle [6]. Animațiile de mers și shooting sunt preluate de pe Mixamo, un tool online gratis ce conține o bază de date cu animații generice pentru umanoizi. Pentru trecerea cursivă de la o animație la alta (i.e. de la mersul înainte la mersul în lateral) sunt folosiți blend trees ce interpolează între mai multe animații în funcție de doi parametrii: viteza pe axa oZ și viteza pe axa oX. Informațiile necesare pentru acest proces au fost preluate din [11], cap. 8 + tutoriale online.

Deplasarea inamicilor este realizată cu ajutorul unui NavMesh ce calculează aria “walkable” la runtime, după ce terenul a fost generat. Aceștia încep să urmărească player-ul când el intră în raza lor de detectare (player-ul poate fi detectat chiar daca se află în spatele inamicului), în schimb, inamicii nu încep să tragă decât dacă player-ul se află în view range (în formă de con, iar raza de acțiune este mai mică decât cea folosită pentru detectare). Informațiile necesare pentru comportamentul inamicilor au fost preluate din [11], cap. 10 și tutorialele lui Brackeys [12]. Codul ce permite modificarea view range-ului unui inamic (ce extinde funcționalitatea editorului Unity) a fost preluat din [11] (clasa EnemyVisibilityEditor, fișierul EnemyVisibility.cs).

Animația de atac a fost adăugată pe un layer diferit în Animator Controller și are aplicat un AvatarMask doar pentru partea superioară a corpului, astfel inamicul poate trage și se poate deplasa în același timp (Animații diferite pentru atac și mers. În animația de atac, inamicul este staționar)



(Layer mask-ul aplicat animației de atac. Astfel animația “afectează” doar partea superioară a modelului)

Logica pentru atac este implementată cu ajutorul raycasting-ului. Un script(RifleController) este atașat armei inamicului. Acest script ține o referință către player, iar la fiecare atac, un vector shootDirection este determinat și proiectat prin raycasting pentru a verifica dacă player-ul a fost lovit. Originea focului de atac este reprezentată de către un mic obiect (o sferă) plasat în vârful armei.

Vectorul shootDirection este rezultatul interpolării liniare dintre un vector ce iși are originea în vârful armei și direcția îndreptată către player și vectorul forward al inamicului (astfel inamicul nu poate nimeri player-ul dacă nu este îndreptat spre el). Parametrul t al interpolării este determinat de dificultatea jocului, selectată de către utilizator din meniul de joc.

Animația flamei la momentul atacului este realizată printr-un sistem de particule plasat în vârful armei. De asemenea, animația focului la impact este tot un sistem de particule, poziția acestei animații fiind determinată de către poziția punctului lovit. Sistemul de particule, cel din urmă, este preluat din Standard Unity Asset Pack (este un sistem de particule folosit pentru animarea unor artificii) [8]. Informațiile necesare pentru realizarea animațiilor de atac au fost preluate din tutorialele lui Brackeys [12].

Atacul player-ului este asemănător cu cel al inamicilor. Atât arma player-ului, cât și cea a inamicului au fost preluate din asset pack-ul Sci-Fi Gun-Light [7]. Script-ul atașat armei player-ului este asemănător celui atașat inamicilor, doar că vectorul de atac își are originea în poziția camerei, iar direcția este determinată de vectorul forward al camerei.

**Responsabiliățile fiecărui membru al echipei:**

Dincă Vlad:

* implementarea logicii pentru inamici și logica de atac
* o parte din UI

Istudor Georgiana Cristina:

* Generare labirint

Gavrila Alexandru Ionut

* Miscare player
* Miscare camera
* Sunete
* Logica progresie
* Texturi

Referințe:

<https://assetstore.unity.com/packages/2d/textures-materials/sky/starfield-skybox-92717>

<https://assetstore.unity.com/packages/2d/textures-materials/wood/plank-textures-pbr-72318>

<https://assetstore.unity.com/packages/tools/custom-teleporter-pad-script-20098>

<https://assetstore.unity.com/packages/2d/textures-materials/textures-brick-and-tile-71671>

[Tutorial pentru generare de labirint](https://catlikecoding.com/unity/tutorials/maze/)

<https://assetstore.unity.com/packages/3d/characters/robots/space-robot-kyle-4696>

<https://assetstore.unity.com/packages/3d/props/guns/sci-fi-gun-light-87916>

<https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-for-unity-2017-3-32351>

<https://www.mixamo.com/#/>

Asset pack-ul TextMeshPro ce este integrat în Unity

[Paris Buttfield-Addison, Jon Manning, și Tim Nugent] - Unity Game Development Cookbook

<https://www.youtube.com/user/Brackeys>

<https://gist.github.com/moszeed/2a1bd167fa60556d7099cc2d666cb3db>

<https://www.fesliyanstudios.com/>