**Tema Proiectului**

Proiectul consta in ansamblul unor clase si metode care alcatuiesc un motor de construire pentru jocuri. Am ales sa construiesc un joc in planul 2D, prin urmare (inca) nu ofera support pentru construirea aplicatiilor 3D.

**Functionalitatile care le ofera sunt:**

**- support pentru aplicatia Discord:** posibilitatea de a creea un grup pe Discord, trimitere invitatie si alaturare grupului prin intermediul aplicatiei Discord.  
Obtinerea unui username si imagine de profil in joc, extrase de pe Discord.  
Actualizarea activitatii pe Discord, in functie de stagiul jocului.

**- support pentru aplicatia Steam:** posibilitatea de a creea un grup pe Steam, trimitere invitatie si alaturare grupului prin intermediul aplicatiei Steam.  
Obtinerea unui user-id unic in joc, extrase de pe Steam.  
Actualizarea activitatii pe Steam, in functie de stagiul jocului.

**- support pentru incarcarea si redarea sunetelor:** motorul contine o clasa numita ‘mixer’ care contine un ansamblu de metode pentru a usura utilizarea sunetelor.

**- sistem de shader si rendering** capabil sa genereze un numar foarte mare de cadre fara a observa un impact asupra performantei.

**- incarcare/eliminare imagini sau animatii** ajuta la incarcarea imaginilor in memorie si ofera cateva functii suplimentare pentru scalare, rotatie, plasare, stergere precum si alte functii de alterare a imaginilor prin aplicarea anumitor masti, decupare, inlocuirea culorilor, si aplicarea diferitor valori de transparenta.

**- sistem pentru editare harti** cu utilizare manuala, se poate creea o harta noua, se pot plasa sau sterge obiecte pe aceasta harta, iar sistemul la randul lui poate salva sau incarca o harta din memorie in fisier sau invers.

**- sistem pentru configurarea setarilor**, incarcarea automata dar si salvarea modificarea si incarcarea manuala

- **cateva clase gata construite** pentru ‘bullet’, ‘button’, ‘console’, ‘discord’, ‘entity’, ‘health’, ‘mixer’, ‘particle’, settings’, ‘steam’, ‘text\_widget’, ‘game\_engine’ – acestea contin o parte din metodele necesare pentru construirea jocului

Exemple:

1. **Aplicatia Discord**

Platforma de socializare cu prietenii care ofera si cateva functionalitati pentru jocuri. Folosind Discord SDK putem conecta Discord cu aplicatia noastra



1. **Aplicatia Steam**
2. **Suport pentru incarcare si redarea sunetelor.**

**CLASS** **Mixer(settings)**- constructorul va seta volumul celor doua canale numite ‘sound’ si ‘music’, si va memora setarile primate prin variabila ‘settings’

**mixer.music\_load(path)**  
- incarca un sunet in memorie  
- ‘path’ este un string cu locatia sunetului

**mixer.music\_play(path, repeat\_times, fade\_ms):**- reda sunetul pe canalul ‘music’ incarcat in memorie prin string-ul ‘path’  
- ‘repeat\_times’ reprezinta numarul de cate ori va fi repetat sunetul  
- ‘fade\_ms’ este durata in care sunetul va ajunge de la volumul minim la volumul setat de constructor.

**mixer.music\_stop():**- opreste sunetul de pe canalul ‘music’

**mixer.music\_rewind():**- reda sunetul de pe canalul ‘music’ de la capat

**mixer.music\_pause():**- suspenda redarea sunetului de pe canalul ‘music’

**mixer.sound\_play(**path**):**- continua redarea sunetului specificat in path de pe canalul ‘music’

**mixer.update\_music\_volume():**- actualizeaza volumul pe canalul ‘music’ la noua valoare citita din setari

**mixer.update\_sound\_volume():**- actualizeaza volumul de pe canalul ‘sound’ la noua valoare citita din setari

1. **Sistem de shader si rendering**

- sistemul va citi automat rezolutia ecranului si va incerca sa scaleze ecranul virtual pentru a potrivi rezolutia utilizatorului, fara a pierde rata aspectului  
- Renderul este bazat pe OpenGL folosind vertex si fragment shader.  
- Poate fi aplicat si shader customizat, cum ar fi:

*sample\_pos = vec2(uvs.x + sin(uvs.y \* 0.1 + time \* 0.01), uvs.y);*

- acesta poate altera imagina finala facand ecranul sa se miste pe o  
directive sinusoidala si creeaza un efect aproiat cu miscarea apei.  
- Se pot controla cadrele pe secunda modificand valoarea ‘game\_engine.fps’ sau modificarea fisierului de configurari in sectiunea ‘[VIDEO] fps’  
**game\_engine.surf\_to\_texture(surf):**  
- converteste o imagine de tip ‘Surface’ intr-o textura care poate fi utilizata de placa video, necesar la renderingul prin OpenGL  
- ‘surf’ este un obiect de tip ‘Surface’ si reprezinta aria unei imagini, poate fi generat folosind **pygame.Surface()**

**game\_engine.adjust\_video\_settings(info)**- se ocupa cu scalarea corecta a ecranului virtual pentru a se potrivi cu ecranul utilizatorului.  
- ‘info’ contine datele monitorului si se poate obtine prin **pygame.display.Info()**

**game\_engine.save\_settigns()**- salveaza setarile intr-un fisier de configuratii.

**game\_engine.update\_display()**- ruleaza cateva functii pentru a converti ecranul virtual intr-un context care poate fi folosit de placa video, aplica shaderul si apoi randeaza cadrele pe ecran.

**game\_engine.fade(surf, direction)**- o functie care aplica un efect de fade, in general folosit la tranzitii intre diferite scene.

**game\_engine.fade\_in(surf)**- o functie care aplica un alt tip de fade, in general folosit la tranzitii intre diferite scene.

1. **Functii suplimentare**

In fisierul ‘maintenance.py’ se gaseste o colectie de functii folositoare.  
**notification(font\_size, message, img):**  
- genereaza un mesaj pe ecran

**process\_exists(process\_name):**- verifica in lista de procese windows daca un program ruleaza, folosit pentru a detecta daca Discord sau Steam sunt deschise.

**load\_image(path):**- o functie universala pentru a incarca imagini dintr-o cale specificata

**custom\_mouse():**- folosita pentru a inlocui imagine pointerului (mouse) cu o imagine customizata

**show\_fps(clock, font):**  
- afiseaza in coltul din stanga sus, cate cadre pe secunda sunt generate in joc.