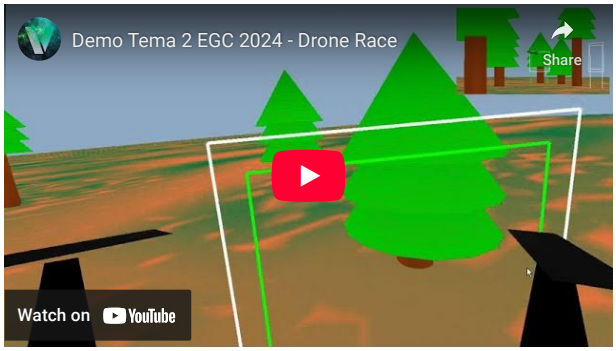
Tema 2 - Drone Mini Games

- **Responsabili:** Anca Bălușoiu, Vlad Novetschi, Robert Caragicu, Silviu Stăncioiu
- **Lansare:** 18 noiembrie 2024
- **Termen de predare:** 17 decembrie 2024, ora 23:59
- **Regulament:** Regulament general
- **Notă: Orice informație ce nu a fost acoperită în acest document este la latitudinea voastră!**

În cadrul temei 2, veți avea de implementat, la alegere, unul dintr-o serie de mini jocuri cu o dronă: race, deliver the package, shooter. Puteți vedea un demo al unui astfel de joc aici:



Demo-ul este pentru jocul de racing, însă nu implementează complet toate functionalitatile, modul de control este ACRO (cu gamepad), viewport-ul din dreapta nu face parte din enunț.

Detalii de Implementare Funcționalități de Bază

Controlul Dronei

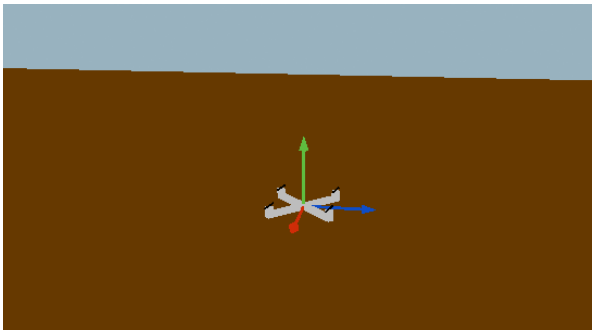
Drona

Drona controlată de jucător va fi formată astfel: un corp format din două paralelipipede așezate în formă de X. În fiecare capăt este atașat un cub unde se găsește câte o elice în formă de paralelipiped. Drona trebuie desenată cu cel puțin două culori distincte: o nuanță de gri pentru corp și negru pentru elice.

Elicele se vor roti cu o viteză constantă în jurul propriului centru și vor rămâne mereu atașate de corpul dronei.

Controlul dronei

Fie animația următoare care ilustrează toate mișcările posibile ale dronei.



În cadrul animației am ilustrat sistemul de coordonate local dronei cu originea în poziția dronei și vectorii de bază formați din săgețile colorate. Ox este săgeata roșie, Oy este săgeata verde iar Oz este săgeata albastră.

Astfel, jucătorul poate controla drona folosind tastele și/sau mouse-ul astfel:

- Translație folosind direcțiile Ox, Oy sau Oz din spațiul local al dronei explicat anterior
- Rotatie doar pe Oy din spațiul local al dronei explicat anterior

Camera

Camera poate fi implementată:

- First-person (din perspectiva dronei, priviand către axa locală OZ / Forward)
- Third-person (poziționată la o mică distanță deasupra și în spatele dronei, privind către direcția de deplasare).

La jocul de livrări este necesară folosirea unei camere third-person pentru a putea observa pachetele atașate de dronă. La celelalte două jocuri, puteți alege varianta preferată.

Generare Hartă și Obstacole

Scena

Info curs
- Elemente de Grafică pe Calculator

Cataloage EGC
- Notele vor fi vizibile pe Moodle individual

Laboratoare
- Setup framework
- Laboratorul 01
- Laboratorul 02
- Laboratorul 03
- Laboratorul 04
- Laboratorul 05
- Laboratorul 06
- Laboratorul 07
- Laboratorul 08
- Laboratorul 09

- Resurse: Redare text
- Resurse: Modele proprii

Teme
- Regulament general
- Calendar Teme

- Tema 1 - Tank Wars
- Tema 2 - Drone Minigames
- Tema 3 - Cappadocian Balloons

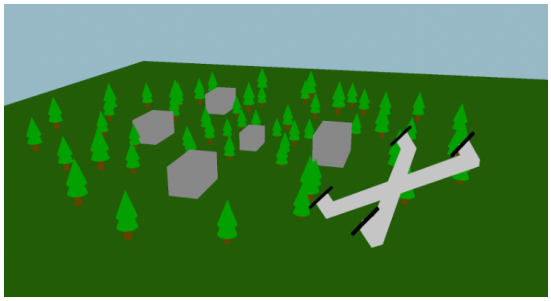
Resurse
- Resurse Utile
- Notare

Table of Contents
- Tema 2 - Drone Mini Games
  - Detalii de Implementare Funcționalități de Bază
    - Controlul Dronei
    - Camera
    - Generare Hartă și Obstacole
    - Detectarea Coliziunilor
  - Detalii de Implementare Funcționalități Avansate
    - Modul Simulator pentru Controlul Dronei
    - UI
    - Shadere
    - Minigame 1: Joc de Curse
    - Minigame 2: Joc de Livrări
    - Minigame 3: Joc Shooter
  - Barem
    - Funcționalități de Bază (150p)
    - Funcționalități Avansate (75p)
  - Exemple de Funcționalități Bonus
  - Întrebări și Răspunsuri
  - Notare
  - Indicații Suplimentare
  - Arhivarea Proiectului

Scena va conține ca elemente de bază:

- Drona controlată de jucător
- Un plan ce reprezintă terenul
- Cel putin un tip de obstacole pe care jucătorul trebuie să îl evite. Obstacolele pot fi orice obiecte va doriti însă aveți grijă să puteți respecta cerința legată de coliziuni (Cel putin unul din tipurile de obstacole trebuie sa fie format din 2 corpuri geometrice). De exemplu:
  - Copaci formați dintr-un cilindru maro ce reprezintă trunchiul și două conuri așezate ca în figură ce reprezinta frunzele
  - Clădiri reprezentate prin paralelipipede (+ un acoperis in forma de prisma daca este singurul tip de obstacole).

Obstacolele pot fi de diferite mărimi, pentru a evita un aspect uniform al scenei.

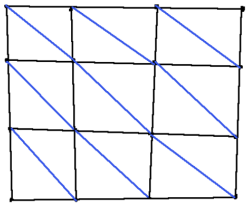


La fiecare pornire a jocului se vor amplasa aleator cel puțin 5 obstacole astfel încât acestea să nu se suprapună.

Teren

Pentru teren se va genera un mesh cu formă de grid regulat, având rezoluția rezoluția mn. Grid-ul trebuie să fie format din dreptunghiuri (alcătuite din 2 triunghiuri) și să nu aibă vertecși duplicați.

Exemplu de grid de 3×3:



Acest mesh va fi generat în C++ folosind o listă de vertecși și una de indici, precum în laboratorul 2. Toți vertecșii vor fi generați având componenta y a poziției egală cu 0, obținându-se un grid drept. Culorile și normalele atribuite în crearea mesh-ului nu au importanță, puteți alege valori constante pentru acestea.

Pentru a-i oferi terenului o formă și o aparență plăcută, în timpul randării, se va modifica înălțimea vertecșilor acestuia (componenta y a poziției în Object Space) în Vertex Shader. În Fragment Shader se vor atribui și culori diferite vertecșilor pentru a obține un efect de textură.

O modalitate simplă pentru a face această modificare a înălțimilor este folosirea de noise: https://thebookofshaders.com/11/ .

Se apelează o funcție de noise care primește ca argument o variabilă de tip vec2 care reprezintă poziția în spațiu pentru care se face sampling la noise. Această valoare poate fi poziția (x, z) a vertexului în Object Space, eventual înmulțită cu o constantă (frecvența).

Noise-ul returnează o valoare cuprinsă între 0 și 1 (sau între -1 și 1 în funcție de implementare), valoare care poate fi folosită pentru a modifica înălțimea vertexului.

Această valoare poate fi folosită și pentru alegerea culorii unui fragment.

```
final_color = mix(culoare1, culoare2, valoare_noise);
```

Detectarea Coliziunilor

Drona nu va putea trece prin obstacole și nici prin sol. Astfel, pentru a implementa această funcționalitate va fi nevoie să realizați coliziunea dronă - sol și dronă - obstacol. Mai mult, în funcție de jocul implementat, va fi nevoie să implementați coliziunea dronă - checkpoint, dronă - pachet sau dronă - proiectil.

Pentru coliziuni, puteți aproxima obiectele astfel:

- Drona va fi un punct sau o sferă
- Obstacolele vor fi construite din minim 2 corpuri geometrice diferite la alegere dintre: paralelipiped aliniat cu axele, sferă, cilindru, con, prisma cu baza aliniata cu axele, capsulă.
- Checkpoint-urile pot fi de mai multe feluri, fie folosind unul din corupurile geometrice de mai sus, fie detectand daca drona a trecut de o parte sau de alta a unui patrat / cerc (ca in demo)
- Pachetul va fi un paralelipiped aliniat cu axele
- Proiectilul va fi o sferă

Mai multe informatii despre coliziuni si cum se pot implementa în 3D: https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection

Detalii de Implementare Funcționalități Avansate

Modul Simulator pentru Controlul Dronei

Pentru cerințele avansate, puteți opta să implementați un mod de simulare pentru controlul realist al dronei.

În realitate, toate motoarele unei drone sunt paralele, ceea ce înseamnă că drona poate genera accelerație directă doar pe axa locală Oy. Totuși, prin ajustarea puterii motoarelor, drona poate să se rotească în jurul oricăreia dintre cele trei axe locale.

Pentru controlul avansat, va fi necesar să implementați:

- Simularea gravitației și a forței de propulsie generată pe axa Oy.
- Un sistem de control pentru unghiul dronei, care să permită manipularea direcției de mișcare a acesteia.

Controlul Avansat: Input-ul Utilizatorului

Utilizatorul va controla drona folosind următorii parametri:

- **Thrust** (forța de propulsie),
- **Pitch** (înclinarea sus-jos în jurul axei locale Ox),
- **Yaw** (rotirea stânga-dreapta în jurul axei locale Oy),
- **Roll** (rotația trigonometrică/anti trigonometrică în jurul axei locale Oz).

Recomandări pentru Taste:

- **Thrust**: tastele W/S.
- **Pitch**: săgețile sus/jos.
- **Yaw**: tastele A/D.
- **Roll**: săgețile stânga/dreapta.

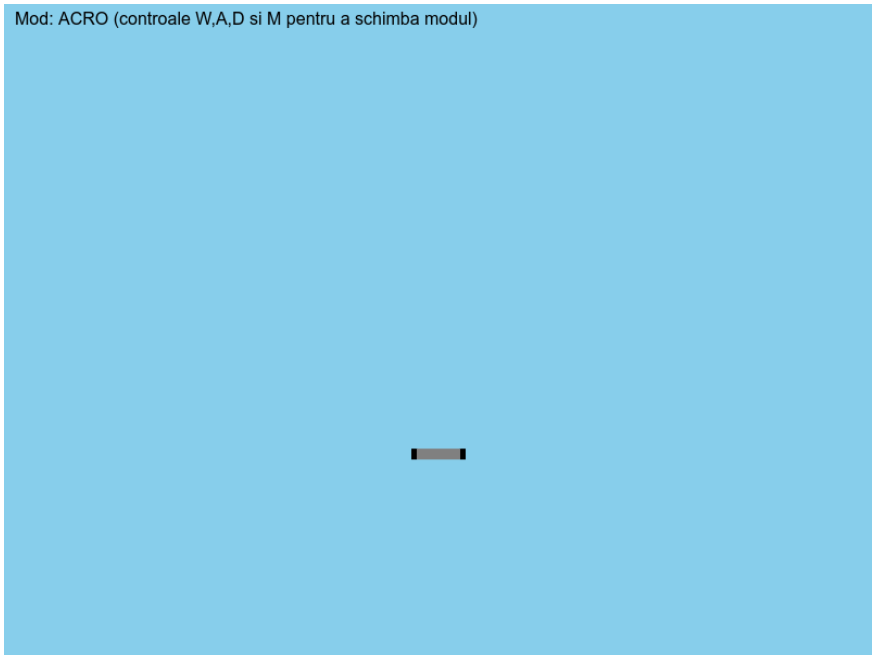
Alte recomandari pentru tastele / input-urile folosite pot fi: * Shift/Z W/S Q/E A/D (Thrust Pitch Yaw Roll), care ar permite si utilizarea mouse-ului pentru o alta functionalitate (ex tragerea in alta directie decat centrul ecranului) * Utilizarea mouse-ului pentru pitch si yaw sau roll (pentru mai multa precizie).

Metode de Traducere a Input-ului Utilizatorului în Unghiurile Dronei

Există mai multe moduri de a traduce input-ul utilizatorului pentru a controla unghiul dronei:

1. Modul "Angle": Utilizatorul controlează direct unghiul dronei față de orizont. Apăsând Sus/Jos, drona se va înclina până la -15/+15 grade față de linia orizontului pe axa locală Ox. Săgețile stânga/dreapta vor controla în mod similar rotația față de axa Oz. Tastele A/D vor ajusta viteza de rotație față de Oy.
2. Modul "Acro": Utilizatorul controlează viteza de rotație a dronei. La apăsarea tastelor Sus/Jos, drona va începe să se rotească în jurul axei Ox cu o anumita viteză unghiulară. Viteza poate fi constantă sau poate crește treptat, în funcție de durata de apăsare a tastei. În acest mod, drona va putea efectua manevre acrobatice precum flip-uri.

Cele două moduri sunt ilustrate în 2D în această demonstrație:



Dacă observați ca menținerea altitudinii este dificilă in acest mod, puteți opta (ca funcționalitate bonus) pentru implementarea unui controller pid pentru altitudine.

Dacă dețineți / preferați gamepad, îl puteți folosii ca metodă de input cu:

```
if (glfwJoystickPresent(0))
{
    int count;
    const float* axes = glfwGetJoystickAxes(0, &count);
}
```

Însă este necesar să permiteți și input din taste, iar suportul pentru gamepad-uri nu se consideră ca fiind functionalitate bonus/nu se vor primi puncte în plus. (Cu toate acestea, poate face jocul mai distractiv 🙂)

UI

Implementarea Minimap-ului

Pentru implementarea unui minimap, cea mai simplă variantă este ca, după ce ați desenat restul scenei, să curățați bufferul de adâncime (depth buffer) și să randati din nou scena folosind o cameră ortografică poziționată deasupra dronei, mutând viewport-ul într-un colț al ecranului. Pentru a face mai vizibile anumite elemente (drona, checkpoint-urile, inamicii, pachetele, destinația, indicatoarele), le puteți randati diferit: de exemplu, drona ca o săgeată, iar checkpoint-urile ca linii cu o grosime mai mare, setate prin **glLineWidth**.

Implementarea Split Screen

Pentru a implementa split screen, va trebui să păstrați datele pentru două sau mai multe drone. Perspectiva fiecărui jucător va fi randată folosind un viewport dedicat: pentru primul jucător, viewport-ul va fi plasat în jumătatea stângă a ecranului, iar pentru cel de-al doilea jucător, se va folosi o altă cameră și viewport-ul va fi setat în jumătatea dreaptă a ecranului. Fiecare jucător va avea taste de control diferite.

În funcție de tipul de joc:

- Jocul de curse: Timpul și checkpointurile parcurse vor fi contorizate separat pentru fiecare jucător.
- Jocul de livrări: Fiecare jucător va avea mai multe pachete de livrat, dar vor fi si pachete ce pot fi livrate de ambii jucători.
- Jocul shooter: Ambii jucători vor putea trage în inamici, iar, opțional, se poate implementa și un mod PvP.

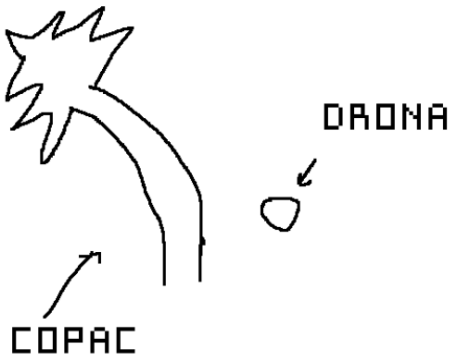
Shadere

1. În funcție de altitudinea / distanța la care se află jucătorul, terenul și obiectele din scenă își schimbă culoarea. Se poate face o interpolare simplă între două culori în fragment shader în felul următor (aceasta este doar o sugestie de implementare, puteți experimenta și cu alte efecte asemănătoare):
   ```
   out_color = mix(culoare1, culoare2, clamp(altitutine_player / altitudine_maxima, 0, 1))
   ```
2. Dithering: Efectul de dithering presupune folosirea unei palete de culori reduse și aproximarea culorilor adevărate folosind culori din paleta redusă pentru pixeli adiacenți. Exemplu imaginea următoare:
   
   Dacă îi reducem paleta de culori, aceasta arată așa:
   
   Iar aplicând un algoritm de dithering care folosește combinații ale pixelilor adiacenți pentru a aproxima culorile inițiale folosind culorile din paleta redusă obținem următorul rezultat:
   
   Există mulți algoritmi de dithering, dar nu toți sunt ușor de implementat pe placa grafică. Un algoritm de dithering care poate fi implementat în Fragment Shader este următorul: Alex Charlton — Dithering on the GPU . Se definește un mask, iar în funcția de poziția pixelului pe ecran și valoarea din mask aferentă acelui pixel se alege fie cea mai apropiată valoare din paleta de culori redusă, fie următoarea culoare, efectul fiind unul convingător. Este de menționat faptul că dithering-ul se aplică la nivel de pixel, deci calculele se efectuează pozițiile pe ecran ale pixelilor (gl_FragCoord.xy).
3. Umbră: Shader care randează umbra unui obiect. Acest shader înainte de a transforma pozițiile vertecșilor din Object space în World Space, realizează o proiecție a acestora pe planul XZ (de exemplu: se poate efectua o scalare pe OY cu un factor foarte mic, aproape 0). Matricea de modelare folosită pentru acest model trebuie modificată astfel încât y-ul în World Space al obiectului proiectat să fie deasupra terenului. Pentru a nu apărea probleme la suprapunerea cu terenul, setați un bias (ex: 0.1) al înălțimii astfel încât umbra să fie puțin deasupra denivelărilor terenului. De asemenea, puteți schimba forma umbrei ca aceasta să nu mai fie plană pentru și a se mula pe suprafața terenului (efectuați fix aceeași transformare pe care o efectuați și în cazul terenului cand setati înălțimile vertecșilor). Obiectul va fi randat cu negru pentru a da impresia de umbră.
4. Îndoire a copacilor: Când drona jucătorului este în apropierea copacilor aceștia trebuie să se îndoaie în direcția opusă față de dronă. Sugestie de implementare în Vertex Shader:
   a. Se calculează distanța dintre dronă și centrul copacului (în planul XZ, nu în 3D).
   b. Dacă această distanță este mai mică decât o rază definită r_bend, atunci se calculează un factor de îndoire al copacilor f = clamp(1 - (distanta / r_bend), 0, 1)
   c. Factorul descris anterior este înmulțit cu un unghi maxim de rotație în jurul trunchiului, și cu un alt factor care are valori între 0 și 1 de-alungul înălțimii copacului (0 la bază, 1 în vârf). Se obține

unghiul de rotație al vertexului curent.

d. Se calculează axa de rotație perpendiculară pe direcția dintre dronă și copac (se proiectează această direcție pe planul XZ și se realizează produs vectorial între această axă și axa globală OY).
e. Se efectuează în shader rotația vertexului folosind axa și unghiul de rotație calculate anterior.

Folosind această implementare rezultatul ar trebui să fie următorul:



Minigame 1: Joc de Curse

În acest joc obiectivul dronei este să se deplaseze cât mai repede printr-o serie de porți. **Porțile trebuie să fie parcurse într-o ordine stabilită la începutul jocului.** Porțile trebuie generate aleator similar cu obstacolele (fără ca acestea să se intersecteze cu alte porți sau cu obstacolele). Traseul se va stabili ca o permutare aleatoare sau prestabilita a porților.

Fiecare poartă va fi reprezentată printr-un model 3D. În timpul jocului se va desena cu o culoare distinctă poarta care este următoarea din traseu care trebuie parcursă. O poartă se consideră parcursă dacă drona trece prin interiorul ei.

În interfața jucătorului va fi afișată o săgeată care indică direcția unde se află următoarea poartă ce trebuie parcursă.

Există două moduri de joc:

- Un singur jucător. Jucătorul are ca obiectiv să treacă într-un timp limită prin toate porțile. Timpul rămas poate fi afișat în două moduri: folosind o bara similara cu cea de Hp de la tema 1, sau scriind pe ecran folosind RenderText (Resurse bonus).
- Doi jucători în split-screen care fac cursă cine termină primul traseul. Nu există timp limită. La sfârșitul traseului va fi afișat câștigătorul în consolă.

Minigame 2: Joc de Livrări

Scopul acestui joc va fi acela de a prelua pachete, pe care trebuie să le duceți la o destinație de pe hartă. Pachetul va fi reprezentat de o cutie, care se va instanția la o poziție aleatoare pe hartă. Drona va prelua pachetul când se află suficient de aproape de pachet (intră în coliziune cu pachetul), iar apoi destinația va deveni vizibilă pe hartă. Destinația se va genera la o poziție aleatoare și va randată sub forma unui cerc pe sol.

Drona va duce pachetul către destinație, pachetul fiind în permanență plasat sub dronă de-a lungul traseului. Pentru a putea observa pachetul plasat sub dronă, acest joc va fi third person.

Odată ajunsă la destinație (drona se află în interiorul cercului destinație, dar nu neapărat pe sol; deci nu neapărat la aceiași înălțime cu cercul destinație), pachetul se consideră livrat, desprins de dronă și se va instanția un pachet la o nouă destinație.

Pentru a ști unde trebuie livrat pachetul tocmai colectat, jucătorul va avea la dispoziție un minimap, unde va putea vedea toată harta pe unde se poate plimba. Mai mult, pe sol va fi afișată în permanență o săgeată cu direcția în care se află destinația.

Scopul jocului va fi adunarea cât mai multor pachete. În acest sens, jucătorul va putea vedea în permanență, fie pe ecran, fie în consolă, câte pachete a colectat.

Minigame 3: Joc Shooter

În acest tip de joc există mai mulți inamici amplasați în scenă, de care jucătorul trebuie să se ferească și pe care acesta trebuie să-i elimine.

Mesh-urile inamicilor trebuie să fie generate din cod, având formă de dronă.

Inamicii se vor spawna deasupra marginilor terenului și se vor deplasa cu o viteză constantă spre extremitățile opuse ale hărții. Când ajung în capetele opuse ale hărții, aceștia vor fi despawnați, iar alți inamici se vor spawna în scenă în locul lor.

Inamicii trebuie întotdeauna să fie orientați cu fața către jucător. Acest lucru se poate realiza folosind o rotație în jurul axei OY folosind unghiul de rotație dat de către funcția atan2() care primește ca argument vectorul în planul XZ dintre inamic și jucător (rotit cu 90 de grade pentru ca forward-ul acestuia diferă de poziția (cos(0), sin(0)) de pe cercul trigonometric și inversată componenta z deoarece folosim un sistem de coordonate right-handed). Această operație se numește billboarding.

Este suficient să implementați rotația pe OY, nu este nevoie să-i rotiți și în jurul axei lor right pentru a urmări jucătorul și în sus/ jos.

O dată la un interval de timp vor lansa un proiectil sferic pe direcția jucătorului de care jucătorul trebuie să se ferească. Dacă proiectilul atinge jucătorul (sphere vs player collision detection), se va decrementa viața acestuia (puteți afișa în consolă viața rămasă a jucătorului). Când viața jucătorului ajunge la 0, acesta nu se mai poate deplasa în scenă.

Jucătorul poate ataca inamicii la rândul său. Este la latitudinea voastră dacă sistemul de atac al jucătorului este folosind proiectile sferice precum sistemul inamicilor, sau folosind raycast-uri (intersecții între raze duse pornind de la poziția jucătorului pe direcția de vizualizare a jucătorului și bounding box-urile inamicilor).

Când jucătorul distruge un inamic, se va porni o scurtă animație simplă a inamicului (de exemplu: începe să se rotească și să devină din ce în ce mai mic până dispare complet), iar scorul jucătorului este incrementat (scorul poate fi afișat în consolă).

Barem

Funcționalități de Bază (150p)

- Controlul Dronei în 3D (30p)
  - Mișcare de bază pe axele X, Y și Z (10p)
  - Animatie elice (10)
  - Orientare simplă a dronei (10p)
- Generare Procedurală a Obiectelor (45p)
  - Creare dronă folosind primitive geometrice (15p)
  - Generare procedurală obstacole (20p)
  - Amplasare obstacole (10p)
- Detectare și Răspuns la Coliziuni (40p)
  - Coliziunea cu fiecare dintre cele două tipuri de corpuri generate (15p + 15p)
  - Coliziunea cu planul terenului (XOZ) (10p)
- Teren (35p)
  - Generare teren 5p
  - Deformare din vertex shader (15p)
  - Colorare din fragment shader (15p)

Funcționalități Avansate (75p)

Puteți alege oricare dintre funcționalitățile avansate de mai jos pentru a acumula puncte până la 75. Funcționalitățile suplimentare sunt considerate bonus. Aveți libertatea să alegeți oricare dintre funcționalitățile avansate, fără a fi necesară implementarea tuturor funcționalităților de la un singur tip de joc.

Tipuri de Jocuri (Acestea sunt doar recomandări și pot fi combinate/adaptate după preferință.)

1. **Joc de Curse**
   - Checkpoint-uri/Porți de cursă (15p)
     - Model checkpoint (5p)
     - Detecție coliziune (10p)
   - Cronometru pentru cursă (10p)
   - Split-Screen (20p)
   - Urmatorul checkpoint de culoare diferita (10p)
   - Indicator obiectiv (20p)
2. **Joc de Livrare**
   - Third person camera (5p)
   - Obiecte de colectat și livrat (20p)
     - Coliziune pentru preluare pachet (10p)
     - Detecție desctinație atinsă pentru a lăsa pachetul (5p)
     - Pachetul urmareste drona (5p)
   - UI (45p)
     - Afișarea unui indicator care arată locația punctelor de livrare (5p)
     - Minimap (20p)
     - Sageată cu direcția de mers (20p)
   - Sistem de scor și feedback pentru livrare (5p)
3. **Joc Shooter**
   - Model pentru inamici (10p)
   - Sistem de tragere (30p)
   - Sistem de scor și feedback vizual (animație când moare) pentru fiecare inamic eliminat (15p)
   - Comportament al inamiciilor (20p)
     - Comportament default (10p)
     - Inamicii pot ataca când jucătorul este în apropiere (10p)

Alte functionalitati avansate (doar in cazul ca nu au fost folosite în joculeț)

- Split screen (20p)
- Minimap (20p)
- Indicator obiectiv (săgeată) (20p)
- Modalitati de control avansate (recomandate in special pentru jocul de curse):
  - Modul **ANGLE** (15p): Inputul jucătorului controlează unghiul dronei față de axele principale, precum și forța de propulsie în sus (gravitatie).
  - Modul **ACRO** (20p): Inputul jucătorului controlează viteza unghiulară a dronei față de axele principale. (gravitatie)
- Shadere avansate:
  - **Shader de variație a culorii pe înălțime (10p)**: Terenul sau obiectele își schimbă culoarea în funcție de altitudine, cu o tranziție simplă între două culori.
  - **Shader de dithering** (15p)
  - **Shader de îndoire a copacilor când trece drona** (o deplasare proporțională cu inversul distanței față de centrul dronei) (15p)
  - **Shader simplu de umbră (proiectam toate punctele pe planul XOZ si le randam cu negru)** (15p)
  - **Shader de ceață** : Terenul sau obiectele își schimbă culoarea în funcție de distanță, cu o tranziție simplă între două culori (10p)

Exemple de Funcționalități Bonus

- Nori (10p)
- Oponenti cu AI pentru cursă (30p)
- Sistem avariere dronă la impact (10p)
- Mai multe tipuri de drone (suficient de diferite) (5p)
- Simulare control prin PID

Alte funcționalități pe care nu le-am descris vor fi punctate în concordanță cu dificultatea / calitatea implementării.

Este permisa utilizarea modelelor externe / create in blender / importate *doar* pentru obstacole (si alte obiecte care nu au fost descrise in enunt). Drona si terenul trebuie generate din cod. Coliziunea cu obiectele importate va fi in continuare efectuată asa cum a fost descris in enunt (minim 2 corpuri de coliziune, aceastea trebuie sa fie cat mai apropiate de forma modelului). Este indicat ca daca folositi modele externe, scena sa pastreze o anumita coeziune estetica (daca drona si terenul sunt cu numar mic de poligoane si netexturate, modelele ar trebuii sa aiba un stil similar).

In cazul in care punctajul obtinut din cerințele avansate depășește cele 75p alocate, la bonus se vor adăuga punctele în plus dar cu o valoare de 3 ori mai mica, si se pot obține până la 25 de puncte în plus.

Exemplu 1: daca implementati complet jocul shooter, si adaugati si un mod de joc tip capture the flag care implementează și toate functionalitatile de la jocul de livrari, veți avea punctajul maxim (75 p / 3 = 25p bonus)

Exemplu 2: implementand jocul de curse fără split-screen dar cu cele două moduri de control avansate (- 20p + 15p + 20p) veți obține 15p/3 = 5p bonus.

Întrebări și Răspunsuri

Pentru întrebări vom folosi forumurile de pe moodle. Orice nu este menționat în temă este la latitudinea fiecărui student!

Notare

Baremul este orientativ. Fiecare asistent are o anumită libertate în evaluarea temelor (de exemplu, să dea punctaj parțial pentru implementarea incompletă a unei funcționalități sau să scadă pentru hard coding). Același lucru este valabil atât pentru funcționalitățile obligatorii, cât și pentru bonusuri.

Tema trebuie încărcată pe moodle. Pentru a fi punctată, tema trebuie prezentată la laborator. Vor exista laboratoare speciale de prezentare a temelor (care vor fi anunțate).

Indicații Suplimentare

Tema va fi implementată în OpenGL și C++. Este indicat să folosiți framework-ul și Visual Studio.

Pentru implementarea temei, în folderul **src/lab_m1** puteți crea un nou folder, de exemplu **Tema2**, cu fișierele **Tema2.cpp** și **Tema2.h** (pentru implementare POO, este indicat să aveți și alte fișiere). Pentru a vedea fișierele nou create în Visual Studio în Solution Explorer, apăsați click dreapta pe filtrul **lab_m1** și selectați **Add→New Filter**. După ce creați un nou filtru, de exemplu **Tema2**, dați click dreapta și selectați **Add→Existing Item**. Astfel adăugați toate fișierele din folderul nou creat. În fișierul **lab_list.h** trebuie adăugată și calea către header-ul temei. De exemplu: **#include "lab_m1/Tema2/Tema2.h"**

Arhivarea Proiectului

- În mod normal arhiva trebuie să conțină toate resursele necesare compilării și rulării
- Înainte de a face arhiva asigurați-vă că ați curățat proiectul Visual Studio:
  - Click dreapta pe proiect în **Solution Explorer** → **Clean Solution**
  - Ștergeți folderul **/build/.vs** (dacă nu îl vedeți, **este posibil să fie ascuns**)
- În cazul în care arhiva tot depășește limita de 50MB (nu ar trebui), puteți să ștergeți și folderul **/deps** sau **/assets** întrucât se pot adăuga la testare. Nu este recomandat să faceți acest lucru întrucât îngreunează mult testarea în cazul în care versiunea curentă a bibliotecilor/resurselor diferă de versiunea utilizată la momentul scrierii temei.