❖ Obiective Sprint 2:

In acest sprint am lucrat in echipe de cate trei, fiecare dezvoltand cate 2 minijoculete. Intalnirile au avut loc in zilele de marti, miercuri si joi. In cazul intalnirilor online s-a lucrat cu sharescreen pe teams si
request control.

- ***Implementare nivel puzzle fire*** **DONE**

  - harta

    - obiecte specifice hartii

    - functionalitatea de conectare a firelor

  Obiective noi indeplinite:

  - adaugare de sunete

- ***Implementare nivel puzzle cai*** - S-a decis ca acest minijoc sa fie inlocuit de un joc de pescuit, pentru ca se potriveste maim ult cu tematica jocului.

  - harta X

  -personajele X

  - miscari specifice mutarii cailor (miscari de mouse) X

- ***Implementare nivel puzzle reconstruire plută pentru fluviul fermecat –*** Reconstituire barca

  -harta

  - elementele puzzle (bucatile de ~~pluta~~ barca care trebuie reconstituite)

  - miscari de mouse pentru rotire si unire elemente X - > nu am mai avut nevoie de miscari de mouse pentru rotire si unire elemente, ci doar pentru a muta bucatile de barca la un click de mouse

- ***Implementare nivel labirint*** - **DONE**

  - creare personaj

  - implementare functionalitati de miscare in labirint (dreapta, stanga, jump, fall)

  - implementare calcul numar de pasi facuti in labirint

Obiective noi:

- stabilirea intervalelor pentru obtinerea numarului de stelute (se afiseaza doar in consola)

1. ***<u>Implementare nivel puzzle fire</u>***

Script-urile folosite sunt
Wire.cs si GameLogic.cs

Pentru sunet am folosit funcția PlayClipAtPoint(AudioClip clip, Vector3 position).
Am declarat urmatoarele obiecte de tip AudioClip:
- SuccessSound, pentru cand se conecteaza cu succes toate firele
- SelectSound, pentru cand este selectat un fir
- ConnectSound, pentru cand se conecteaza un fir la altul de aceeasi culoare
- ErrorSound, pentru cand se esueaza conectarea firelor
Acestea pot fi localizate in Assets > Sounds.

Implementare Wire.cs:
-> acest script a fost aplicat pe toate cele 4 fire
-> salvăm poziția inițială a firului în funcția de Start
-> în funcția OnMouseDown setăm boolean-ul Dragging la true și declanșăm sunetul de selectare, întrucât în acest moment player-ul a selectat firul curent.
-> în funcția OnMouseUp setăm Dragging la false și verificăm dacă cele firul selectat a fost conectat cu cel de același culoare de pe partea dreaptă a ecranului. Dacă nu, declanșăm sunetul de eroare și resetăm firul la poziția inițială.
-> în Update, în cazul în care firul este selectat (Dragging = true), preluăm poziția mouse-ului și o setăm ca fiind și poziția in spațiu a firului. Tot aici verificăm și dacă există o diferență mai mică de 0.5f între firul selectat și capătul din dreapta al firului. În caz afirmativ, unim cele două capete ale firului, considerăm cele două fire ca fiind conectate și declanșăm sunetul corespunzător.

Implementare GameLogic.cs:
-> declarăm o listă de Wires, care conține cele 4 fire și o shuffleuim la fiecare început nou de joc. Funcția ShuffleWires este implementată folosind Random, astefel fiind amestecate capetele din dreapata ale firelor ăîn mod aleator.
-> în Update, verificăm fiecare fir dacă este conectat la capătul din dreapta și daca da, creștem un counter de fire conectate. Când acest counter ajunge egal cu numărul de fire din lista Wires, declanșăm sunetul de succes și resetăm jocul cu ResetWires.
-> în funcția ResetWires setăm fiecare fir ca fiind deconectat din nou, astfel modificându-se poziția la cea inițială și apelăm din nou ShuffleWires.

2.Implementare nivel pescuit

Obiectivul jocului este acela de a prinde pestele. Acesta se deplaseaza in sus in jos pe o bara verticala, iar noi trebuie sa il urmarim si sa mutam mouse-ul deasupra lui pana cand bara de progres se umple. Daca luam mouse-ul de pe peste sau daca acesta fuge si nu se mai afla in zona mouse-ului nostru, bara de progres scade, trebuind sa il urmarim constant.

FishingGame - un gameObject care cuprinde toate componentele jocului

Background - un gameObject destinat pentru background

ProgressBarBackground - background-ul destinat scorului cand scade

FishingBackground - apa

fish - un gameObject care afiseaza pestele

Hook - un gameObject care reprezinta hook-ul

ProgressBarContainer - containerul pentru progressBar

ProgressBarFill - background-ul destinat scorului cand creste

TopBounds - un gameObject destinat limitei superioare in care poate ajunge pestele

BottomBounds - un gameObject destinat limitei inferioare in care poate ajunge pestele

Script:FishingMiniGame.cs

private void MoveFish():

functie apelata in FixedUpdate destinata miscarii pestelui. Pestele se deplaseaza in

sus/jos in functie de Random.value. Se foloseste de bottomBounds.position si topBounds.position pentru limite

private void MoveHook():

functie apelata in FixedUpdate destinata miscarii hook-ului. Input.GetMouseButton(0) inregistreaza click stanga, daca acesta este apasat hook-ul se deplaseaza in sus, iar cand nu este apasat, hook-ul se deplaseaza in jos

Mathf.Clamp(hookPosition, hookSize/2, 1-hookSize/2) asigura ca hook-ul nu depaseste

spatiul alocat deplasarii sale

private void CheckProgress():functie destinata scorului

Daca hook-ul se afla sub peste, scorul creste, acesta fiind reprezentat de progressBarContainer.localScale.y.

Daca variabila catchProgress>=1, playerul a castigat, iar daca <=0, acesta pierde

3.Implementare nivel puzzle reconstruire plută pentru fluviul fermecat – Reconstituire barca

Scene - BoatGame

Jocul are ca scop reconstituirea unei imagini cu o barca – imaginea este impartita in 16 parti egale, dintre care o parte este libera.

Scriptul folosit este GameManager.cs

In functia CreateGamePieces se creaza piesele jocului: dimensiunea, pozitia si se initiaza, fiind adaugate in vectorul pieces.

Poza este impartita pe lungime si latime in cate 4 piese (size = 4)

In functia de Update se face shuffle pe partile pozei, la apasare pe mouse, daca piesa selectata este vecina cu piesa goala, atunci aceasta se va face un schimb de pozitii intre cele doua. Verificare ca schimbarea sa fie valida se va face in functia SwapIfValid.

Functia de Shuffle se implementeaza prin brut force.

4. ***Implementare nivel labirint***

- creare personaj

In scena SampleScene:

Personajul este vazut de sus si are forma unei bile albastre (numit Player) si are atasate proprietatile de RigidBody, Circle Collider 2D. Scriptul folosit pentru miscarea personajului este Player_Maze.cs

- implementare functionalitati de miscare in labirint (dreapta, stanga, jump, fall) + implementare calcul numar de pasi facuti in labirint

Scriptul Player_Maze.cs contine pentru fiecare tasta: W,A,S,D functia de transformare a pozitiei jucatorului si in plus Numara si pasii facuti de jucator.

Obiective noi:

- stabilirea intervalelor pentru obtinerea numarului de stelute (se afiseaza doar in consola)

Calupul de if-uri reprezinta cele 3 intervale in care se poate situa jucatorul privind numarul de pasi efectuati:

- [45,54) – 3 stelute

- [54,60) – 2 stelute

- [60,65) – 1 steluta

Daca numarul de pasi este >= 65 atunci jucatorul ori treebuie sa reia jocul (inca nu este implementat) ori daca are un numar de capsune = numarul de pasi efectuati – 65 va putea consuma din capsune si va trece la urmatorul nivel (nu s-a implementat trecerea la urmatorul nivel)

Alte Obiective noi:

- Creare videoclip introducere joc labirint (scena LabirintIntro)

Se creeaza un canvas pe care s-a atasat un video. De asemenea, exista doua butoane: next – care duce jucatorul in scena LabirintMenu si butonul “X” – care duce jucatorul inapoi in jocul de baza – scena MainGame.

- creare meniu joc labirint (scena LabirintMeniu)

Se creeaza un canvas pe care s-a atasat imaginea de fundal, imaginea cu butoanele care o sa fie utilizate in joc si indicatia jocului – indications. De asemenea, exista doua butoane: play – care duce jucatorul in scena SampleScene si butonul “X” – care duce jucatorul inapoi in animatie– scena LabirintIntro.