Paweł Łukaszewski

Portfolio programisty

Snail Hail

4 – 6.03.2022

Unity 2020.3.30f1

Projekt powstał podczas Mage Jam vol V. Temat: "Impact", czas na wykonanie: 48h. Gra jest lokalnym starciem w wyścigach ślimaków.

Zadaniem gracza jest sterując swoim ślimakiem, unikać, albo zbierać deszcz lub grad, które to mają wpływ na prędkość postaci. Wygrywa ten, który jako pierwszy dotrze do domku na końcu trasy.

Kluczowe założenia projektu:

- Ręcznie rysowana, słodka grafika
- Local versus, wyścig
- 2 graczy
- Wybór postaci
- Ślimaki “przyklejone” do podłoża
- Pasująca oprawa dźwiękowa

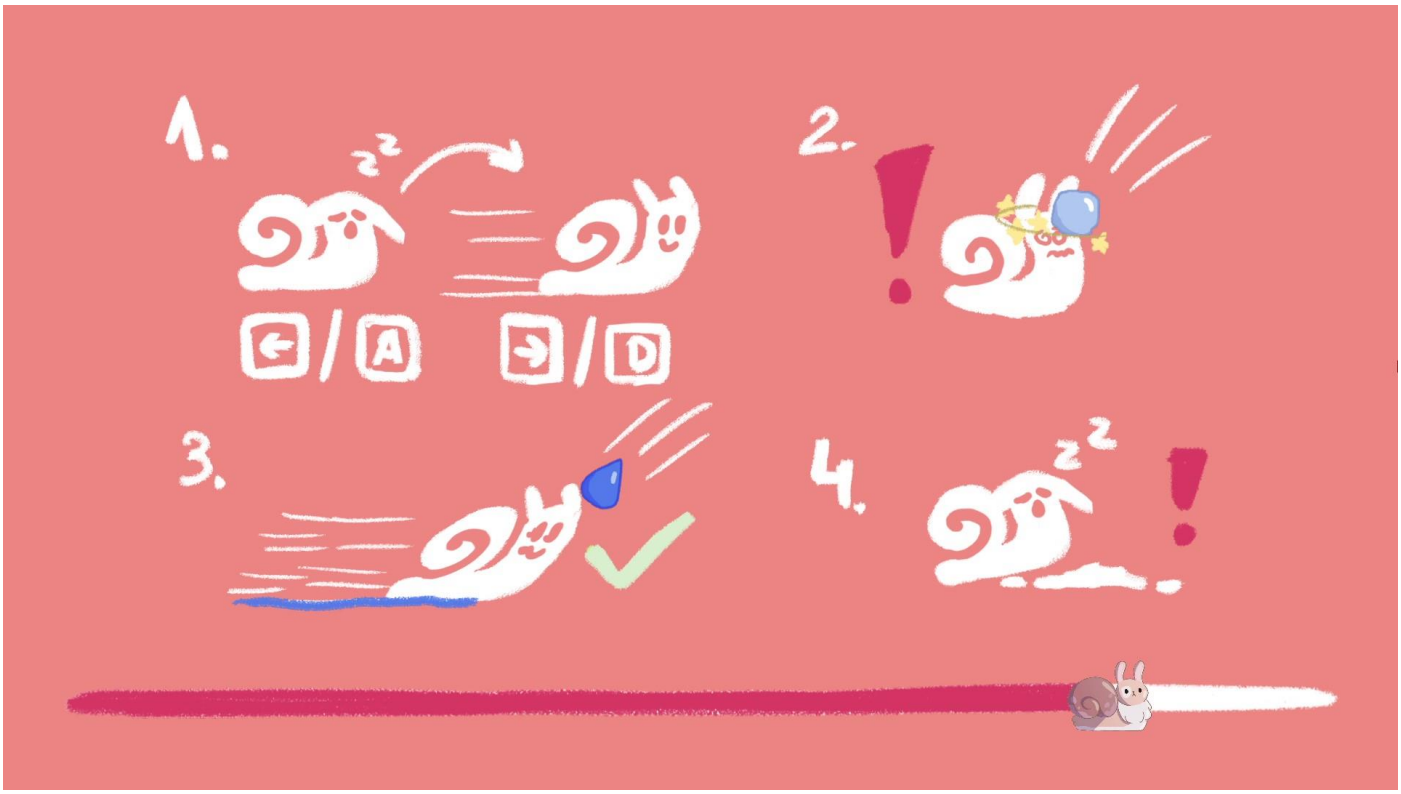
Moja rola w projekcie:

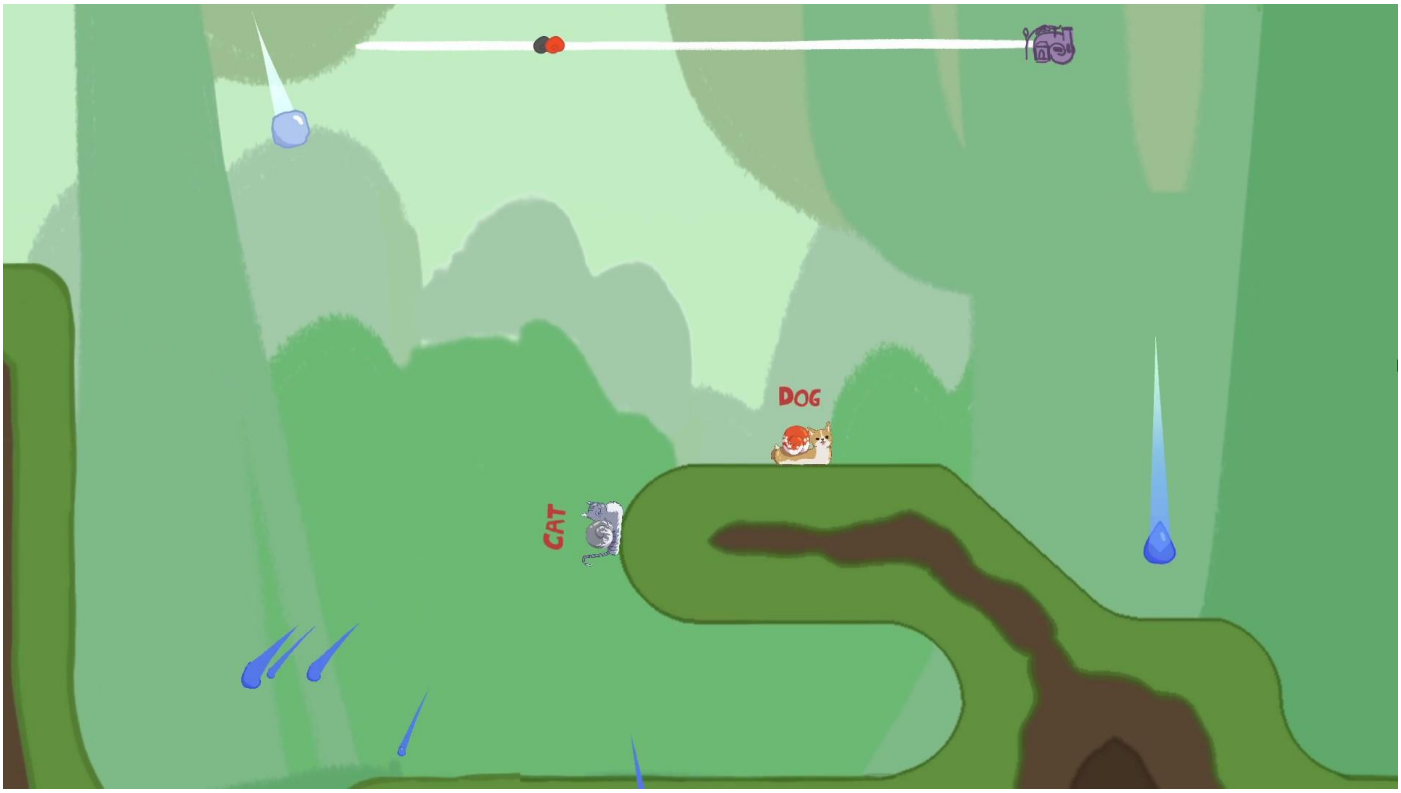
- Programista, odpowiedzialny za zaimplementowanie mechanik

Inne osoby pracujące nad projektem:

- Barbara Kaniewska - graficzka, odpowiedzialna za oprawę graficzną, animacje i UI
- Edyta Janicka – designerka, odpowiedzialna za projekt gry, pomoc przy grafice oraz animacji i udźwiękowienie

Link do Itch.io


1.
A
D
2.
3.
4.


DOG
CAT

Klasa odpowiedzialna za dynamiczne śledzenie graczy poprzez dostosowanie pozycji i rozmiaru kamery.

```
public class CameraController : MonoBehaviour
{
    public GameObject[] followedObjects;
    public float padding;
    public float minimalSize;
    public float yPivot;
    public float lerpRatio;

    private Vector2 maxPos;
    private Vector2 minPos;

    private Camera cam;

    Unity Message | 0 references
    private void Awake()
    {
        cam = GetComponent<Camera>();
    }

    Unity Message | 0 references
    void Update()
    {
        maxPos.x = followedObjects[0].transform.position.x;
        maxPos.y = followedObjects[0].transform.position.y;
        minPos.x = followedObjects[0].transform.position.x;
        minPos.y = followedObjects[0].transform.position.y;

        for(int i = 1; i < followedObjects.Length; i++)
        {
            maxPos.x = Mathf.Max(followedObjects[i].transform.position.x, maxPos.x);
            maxPos.y = Mathf.Max(followedObjects[i].transform.position.y, maxPos.y);
            minPos.x = Mathf.Min(followedObjects[i].transform.position.x, minPos.x);
            minPos.y = Mathf.Min(followedObjects[i].transform.position.y, minPos.y);
        }

        float screenRatio = cam.pixelRect.height / cam.pixelRect.width;
        float halfSize = Mathf.Max(minimalSize, (maxPos.x - minPos.x) * .5f * screenRatio +
          padding, ((maxPos.y - minPos.y) * .5f + padding) / yPivot * .5f);

        cam.orthographicSize = Mathf.Lerp(cam.orthographicSize, halfSize, lerpRatio);
        transform.position = new Vector3((maxPos.x + minPos.x) * .5f, (maxPos.y + minPos.y)
          * .5f + (.5f - yPivot) * halfSize, transform.position.z);
    }
}
```

ProjectArena

03.2021 - 06.2021

Unreal Engine 4.26.2

ProjectArena jest First Person Endless Arena Shooterem. Celem gracza jest przetrwać jak najdłużej w symulacji. Na arenie, na której znajduje się bohater, nieustannie pojawiają się nowe fale przeciwników. Należy być w ciągłym ruchu i rozważnie używać terenu, by nie dać się okrążyć przez nadbiegające ze wszystkich stron boty. Do obrony przed nimi służy karabin automatyczny. Zasoby amunicji, tarczy oraz zdrowia można uzupełnić rozrzuconymi po mapie znajdźkami.

Kluczowe założenia projektu:

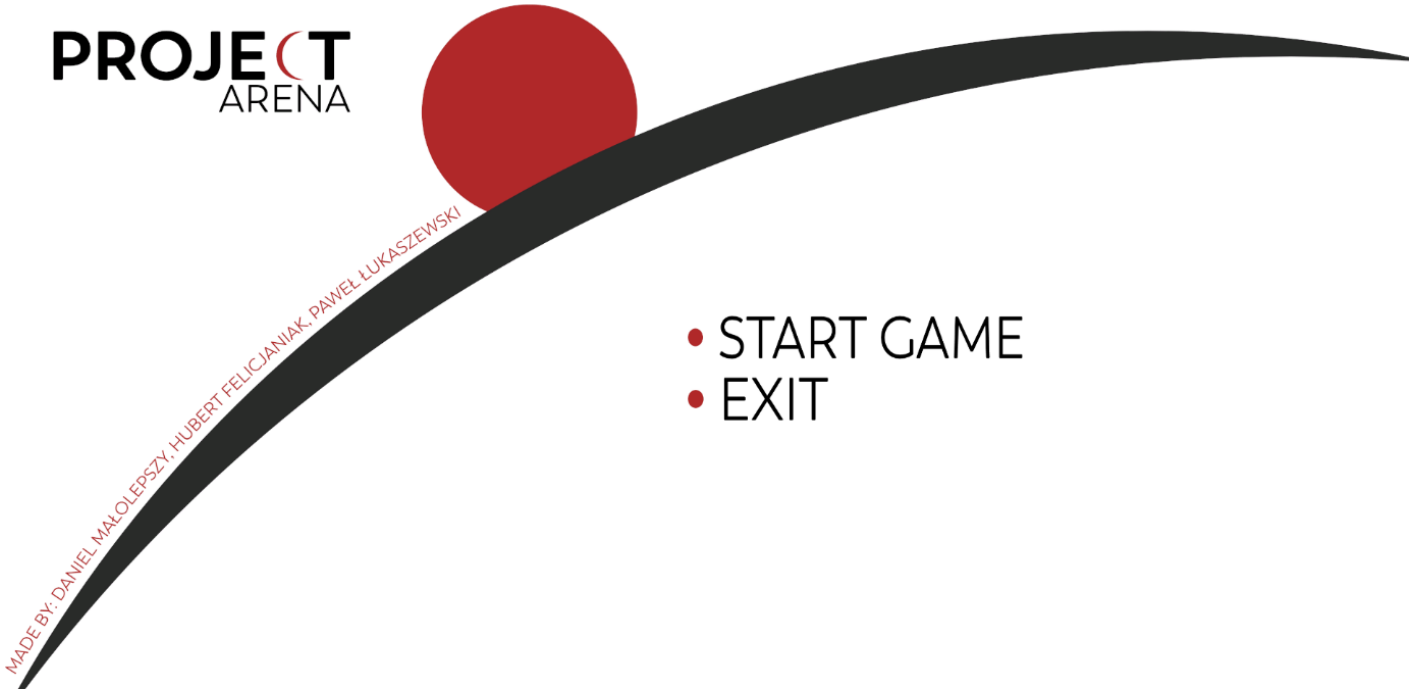
- First Person Endless Arena Shooter
- Niekończące się fale przeciwników
- Proste AI goniące gracza
- Pickupy: życie, amunicja, tarcza

Moja rola w projekcie:

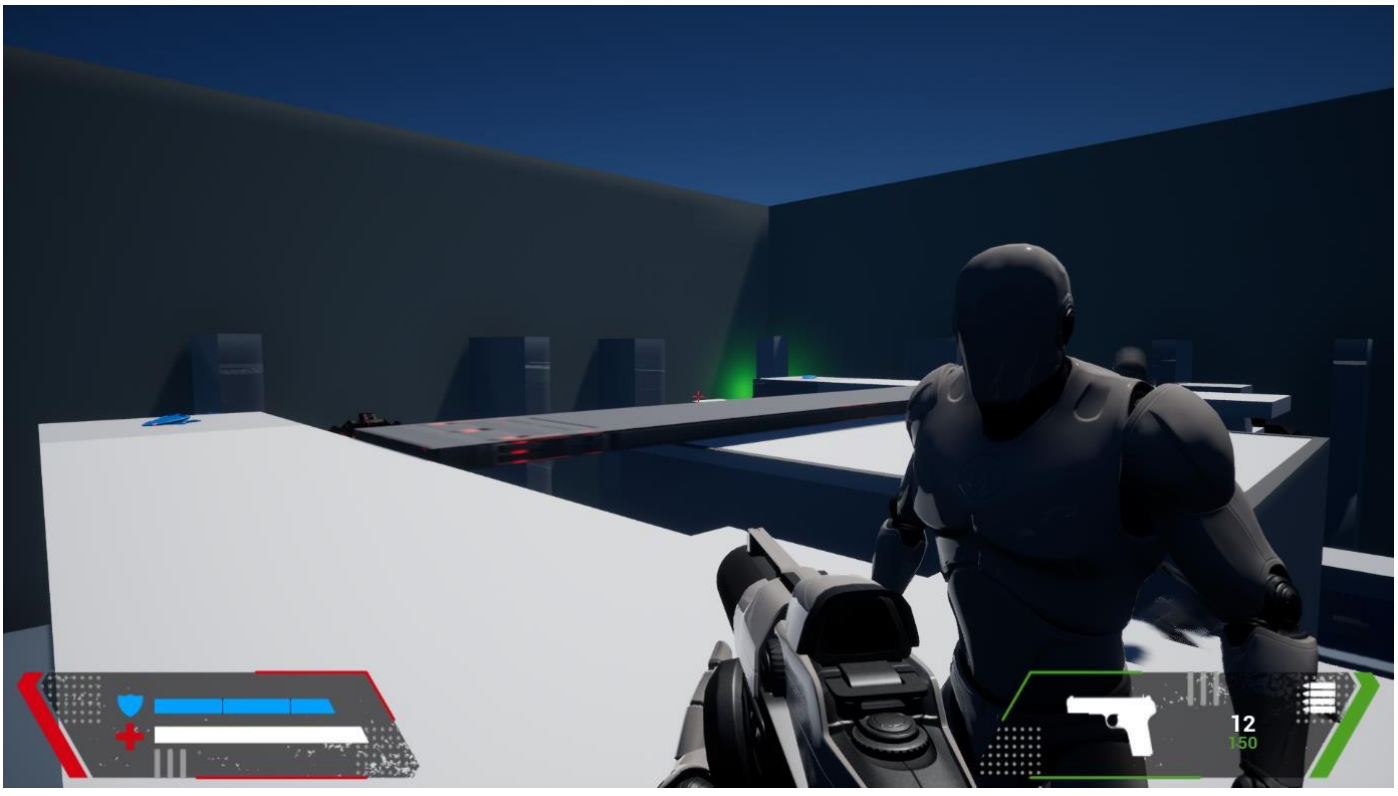
- Programista, odpowiedzialny za zaimplementowanie mechanik (Blueprints), dźwięków oraz UI i zarządzanie projektem

Inne osoby pracujące nad projektem:

- Daniel Małolepszy - grafik/dźwiękowiec, odpowiedzialny za oprawę muzyczną, modele 3D i UI
- Hubert Felicjaniak – level artist, odpowiedzialny za level design i zbudowanie areny




15
150


12
150

ProjectChess

11.2020 - 02.2021

Unity 2019.4.12f1

Projekt powstał jako zaliczenie przedmiotu Projekt Zespołowy na 1. semestrze studiów na Collegium da Vinci na kierunku Informatyka - specjalizacja Projektowanie Gier.

Zadaniem było stworzenie małej gry w temacie "Thinking Ahead".

Gra polega na przemieszczaniu się między pomieszczeniami w celu pokonania przeciwników na szacho-podobnych zasadach.

Kluczowe założenia projektu:

- Pixelart
- Poruszanie się po szachownicy w ściśle określony sposób
- Możliwość zmiany wzoru poruszania się
- Możliwość "zbijania" przeciwników, pokonanie wszystkich przeciwników w pomieszczeniu pozwala na przejście do kolejnego
- Proste AI próbujące "zbić" gracza
- Proceduralnie generowane poziomy z wcześniej przygotowanych pomieszczeń
- Pasująca oprawa dźwiękowa

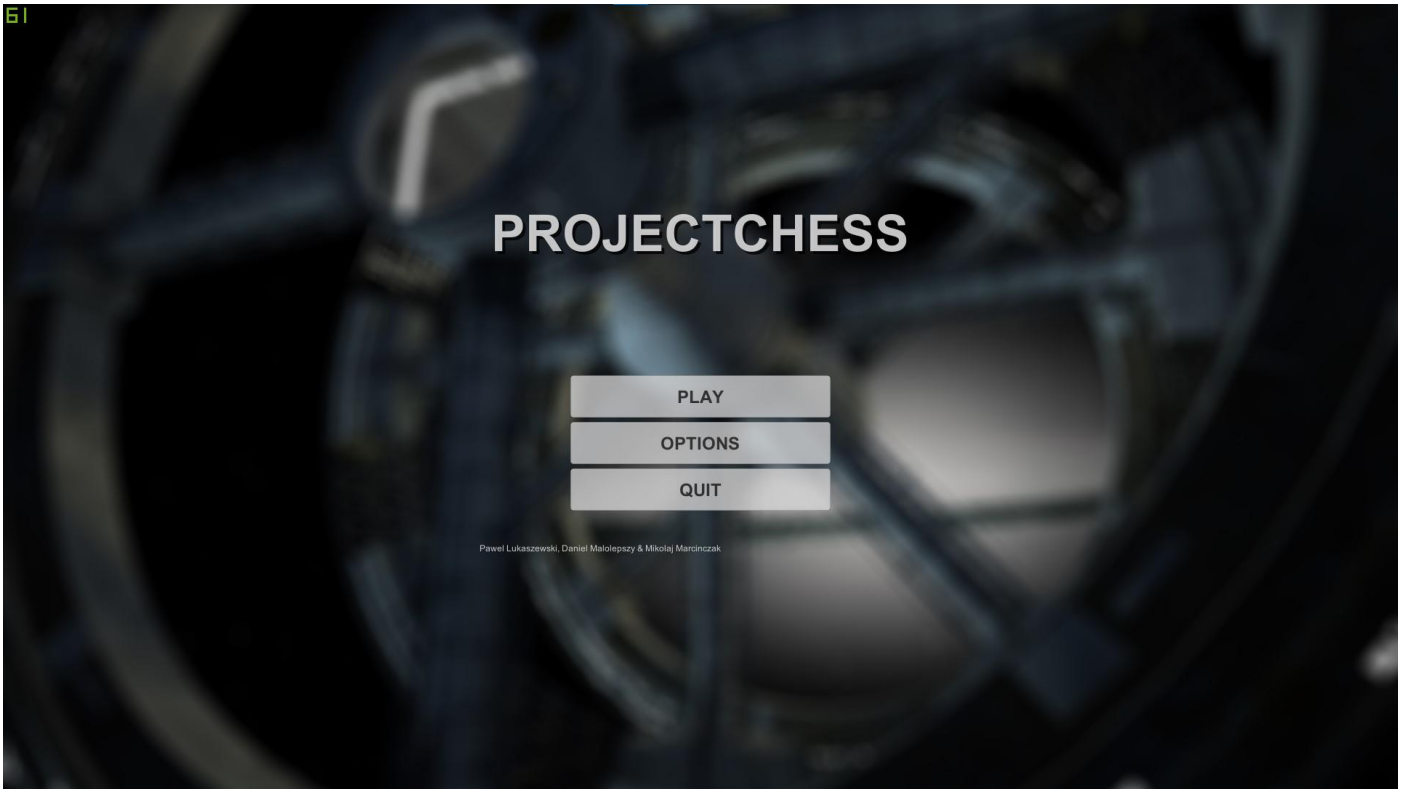
Moja rola w projekcie:

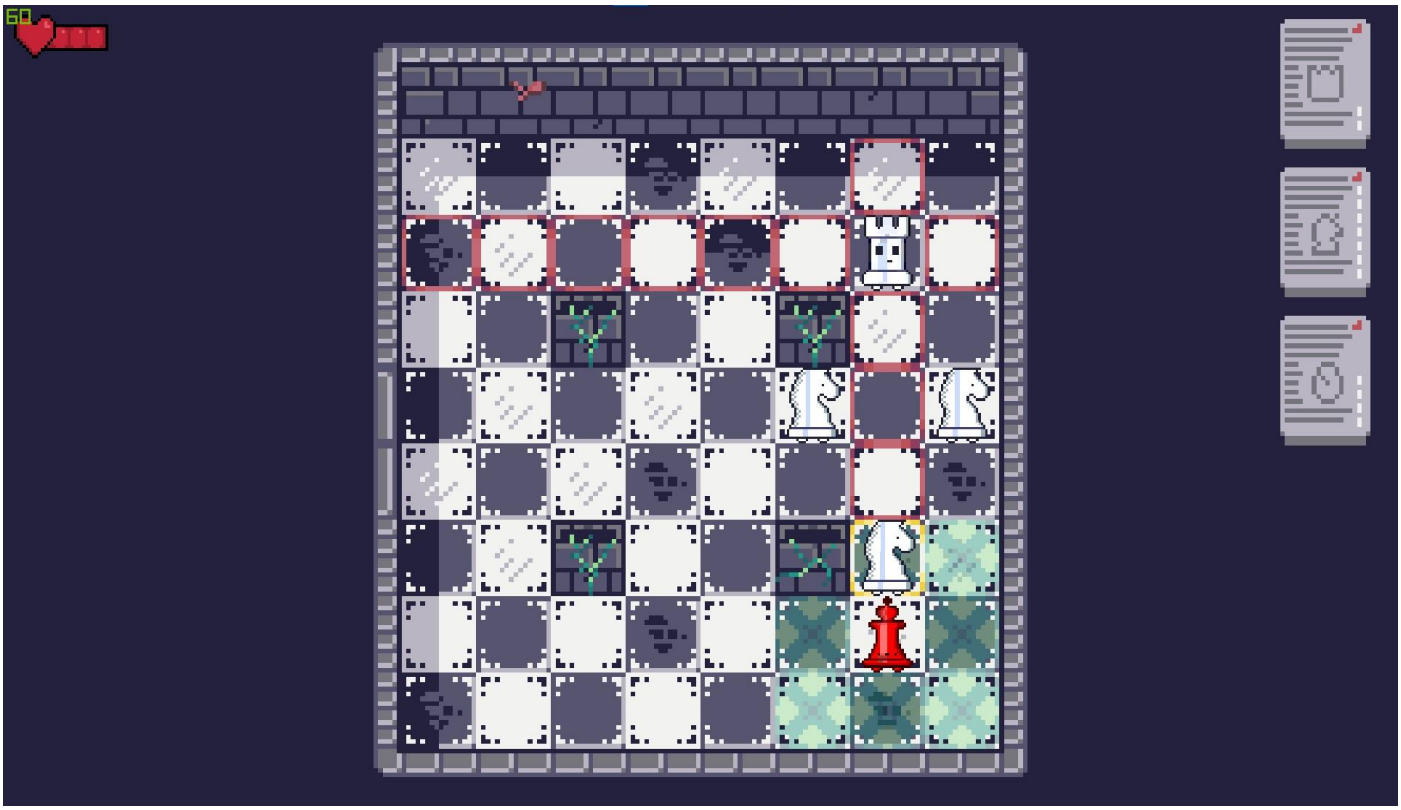
- Programista, odpowiedzialny za zaimplementowanie mechanik i zarządzanie projektem

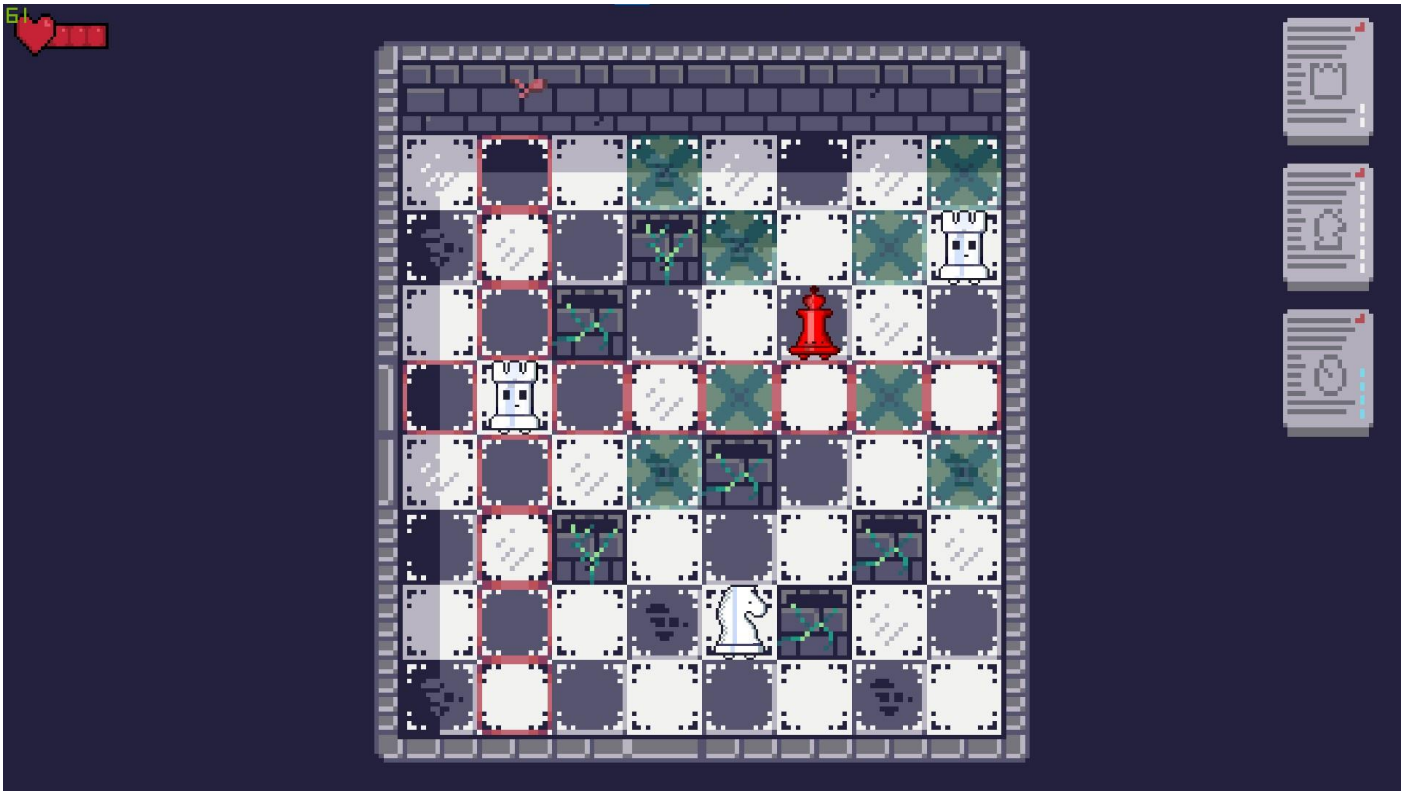
Inne osoby pracujące nad projektem:

- Daniel Małolepszy - grafik, odpowiedzialny za oprawę graficzną i animacje
- Mikołaj Marcińczak - dźwiękowiec, odpowiedzialny za oprawę dźwiękową i UI

Link do Github







Klasa odpowiedzialna za otwieranie/zamykanie drzwi po wyczyszczonym pomieszczeniu oraz umożliwiająca interakcję z nimi.

```
using System.Collections.Generic;
using UnityEngine;

public class Entrance : MonoBehaviour, IClickable
{
    public Vector2Int Direction;
    private bool b = true;
    public GameObject OpenDoors;
    public GameObject ClosedDoors;

    public void Open()
    {
        OpenDoors.SetActive(true);
        ClosedDoors.SetActive(false);
        Global.InputSystem.Subscribe(this);
    }

    public void Close()
    {
        OpenDoors.SetActive(false);
        ClosedDoors.SetActive(true);
        Global.InputSystem.Unsubscribe(this);
    }

    List<Vector2Int> IInputSystemInerface.GetPosition()
    {
        var Tmp = new List<Vector2Int>();

        Tmp.Add(new Vector2Int((int)transform.position.x, (int)transform.position.y));
        Tmp.Add(new Vector2Int((int)transform.position.x + Direction.y, (int)transform.position.y - Direction.x));

        return Tmp;
    }

    void _OnClick()
    {
        Global.GameManager.Level.ChangeRoom(Direction);
        b = true;
    }

    void IClickable.OnClick(int Button)
    {
        if(b)
        {
            Global.GlobalObject.DelayFunction(_OnClick);
            b = false;
        }
    }

    private void OnDestroy()
    {
        Global.InputSystem.Unsubscribe(this);
    }

    public void SetActive(bool b)
    {
        if(b)
        {
            Global.InputSystem.Subscribe(this);
        }
        else
        {
            Global.InputSystem.Unsubscribe(this);
        }
    }

    // Start is called before the first frame update
    void Start()
    {
        Close();
    }
}
```

LoopRunner

5 - 7.03.2021

Unity 2020.1.17f1

Projekt powstał podczas Mage Jam vol 4. Temat: "One screen", czas na wykonanie: 48h. Gra jest endless runnerem.

Zadaniem gracza jest skacząc, utrzymać się na kafelkach z programu muzycznego i przy okazji naprawić melodię. W zespole byłem programistą odpowiedzialnym za implementację mechanik oraz zarządzanie projektem.

Kluczowe założenia projektu:

- Pixelart
- Endless runner
- “Grające” platformy zgrane z muzyką
- Proceduralnie generowane poziomy z wcześniej przygotowanych fragmentów
- Pasująca oprawa dźwiękowa

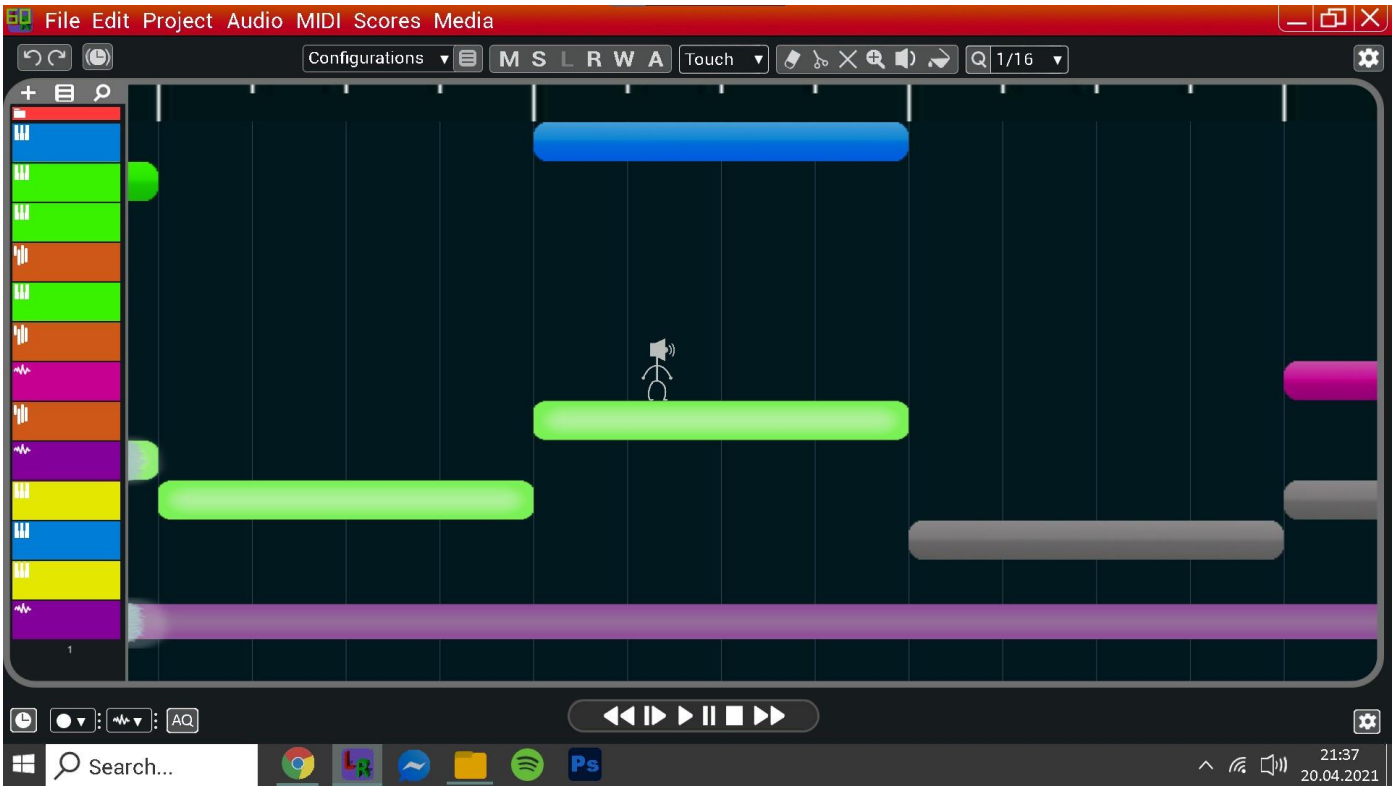
Moja rola w projekcie:

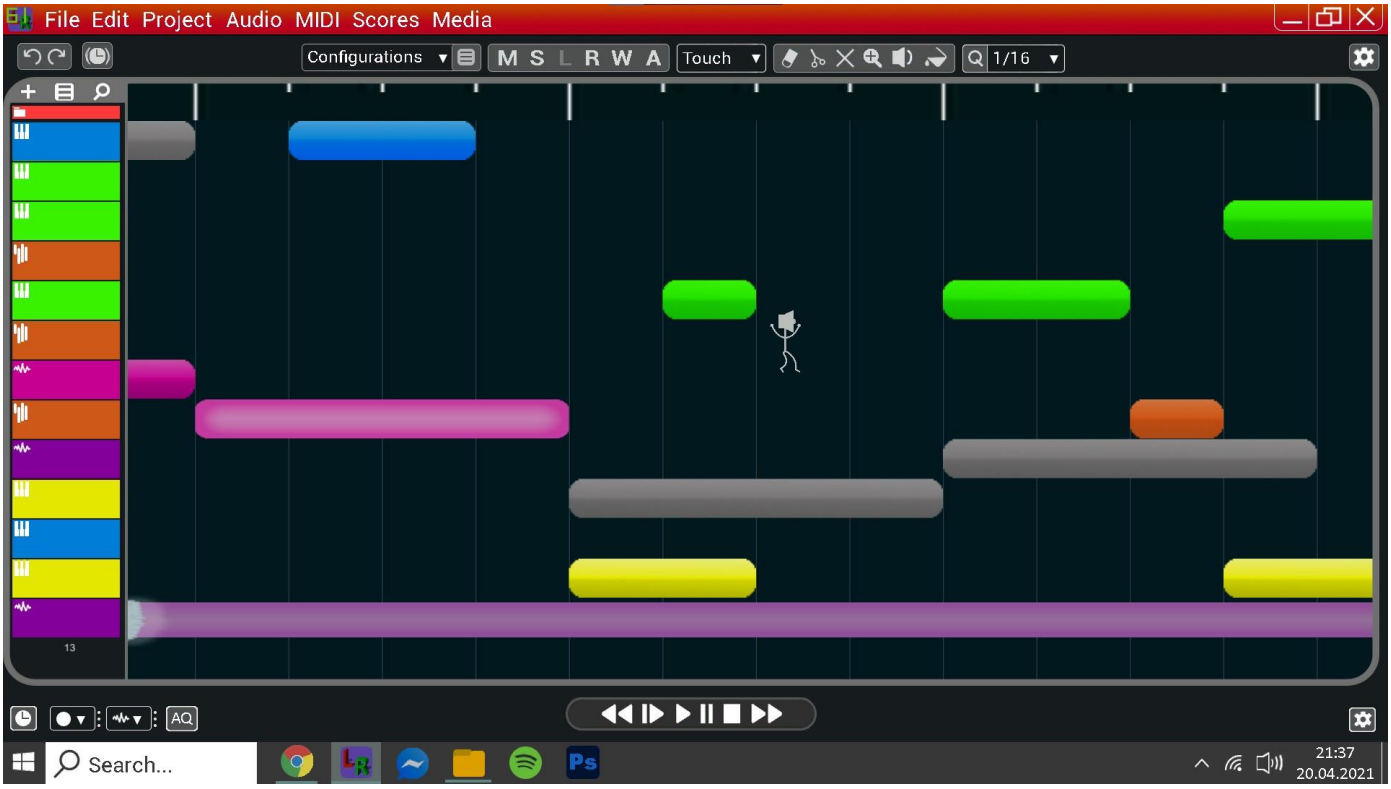
- Programista, odpowiedzialny za zaimplementowanie mechanik i zarządzanie projektem

Inne osoby pracujące nad projektem:

- Michał Szewczuk - grafik, odpowiedzialny za oprawę graficzną, animacje i UI
- Maksymilian Lewiński - dźwiękowiec, odpowiedzialny za oprawę dźwiękową

Link do [Itch.io]


File Edit Project Audio MIDI Scores Media
Configurations
M S L R W A
Touch
1/16
Search...
21:37
20.04.2021


File Edit Project Audio MIDI Scores Media
Configurations
M S L R W A
Touch
1/16
Search...
21:37
20.04.2021

Klasa odpowiedzialna, za generowanie na bieżąco kolejnych segmentów, z których składał się poziom.

```
using System.Collections.Generic;
using UnityEngine;

public class Level : MonoBehaviour
{
    public Segment[] SegmentPrefabs;
    public Segment StartingSegment;
    public List<Segment> Segments;
    public float SegmentWidth = 38.4f;

    private int n = 0;

    public void GenerateNew()
    {
        n++;
        Instantiate(SegmentPrefabs[Random.Range(0, SegmentPrefabs.Length)], new Vector3(n * SegmentWidth,
          0f, 0f) + transform.position, new Quaternion(), transform);
    }

    private void Awake()
    {
        Global.Level = this;

        Instantiate(StartingSegment, transform.position, new Quaternion(), transform);
    }
}
```