Bird Tweeting

16.01.2023

—

Paweł Frankowski
U-19537
3 semestr

Wstęp

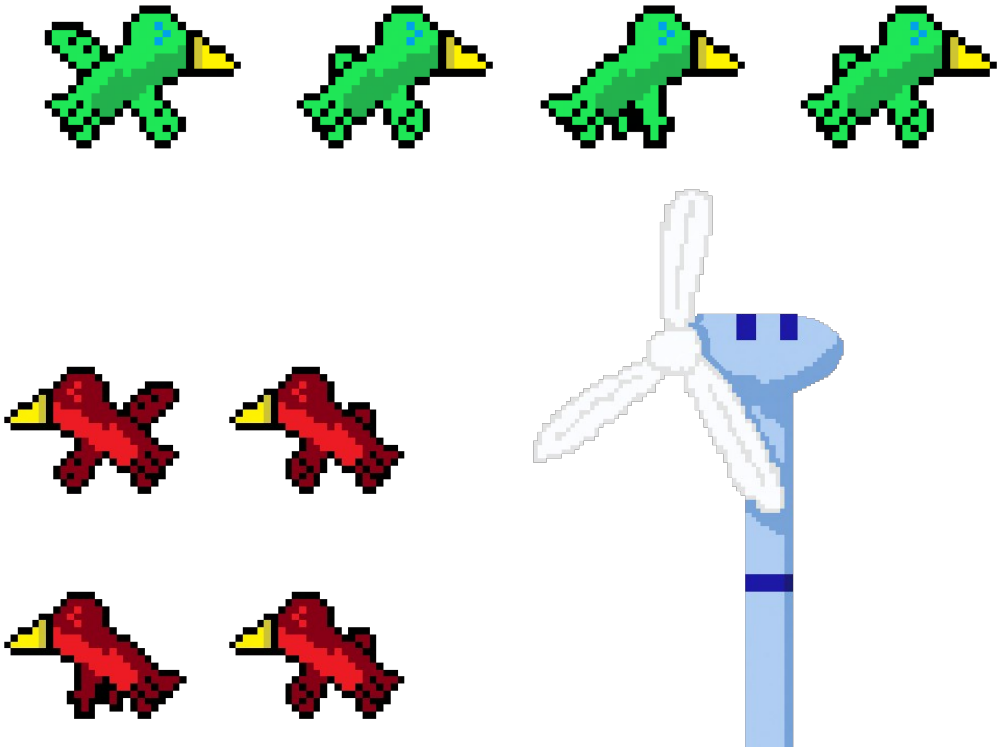
Gra opiera się na zdobywaniu punktów poprzez przelatywanie ptakiem pomiędzy przeszkodami.

Architektura

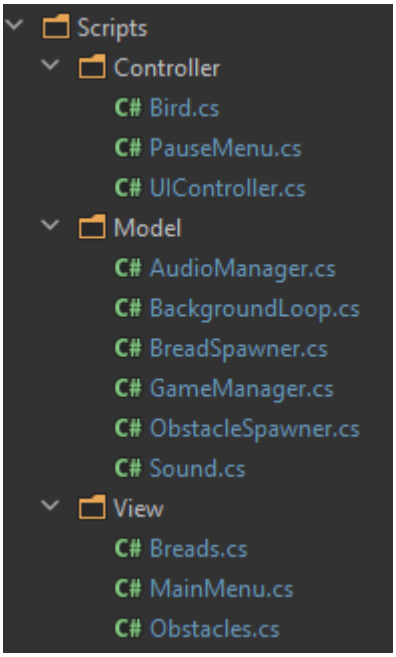
Cała gra opiera się na silniku Unity. Kod źródłowy napisany w 100% w C#.

Tekstury

Tekstury tła, chleba, chmury oraz ziemię pobrałem z strony Unity assets store za darmo. Przeszkody i ptaki zrobiłem samemu na stronie internetowej pixilart.com



Model-view-controller



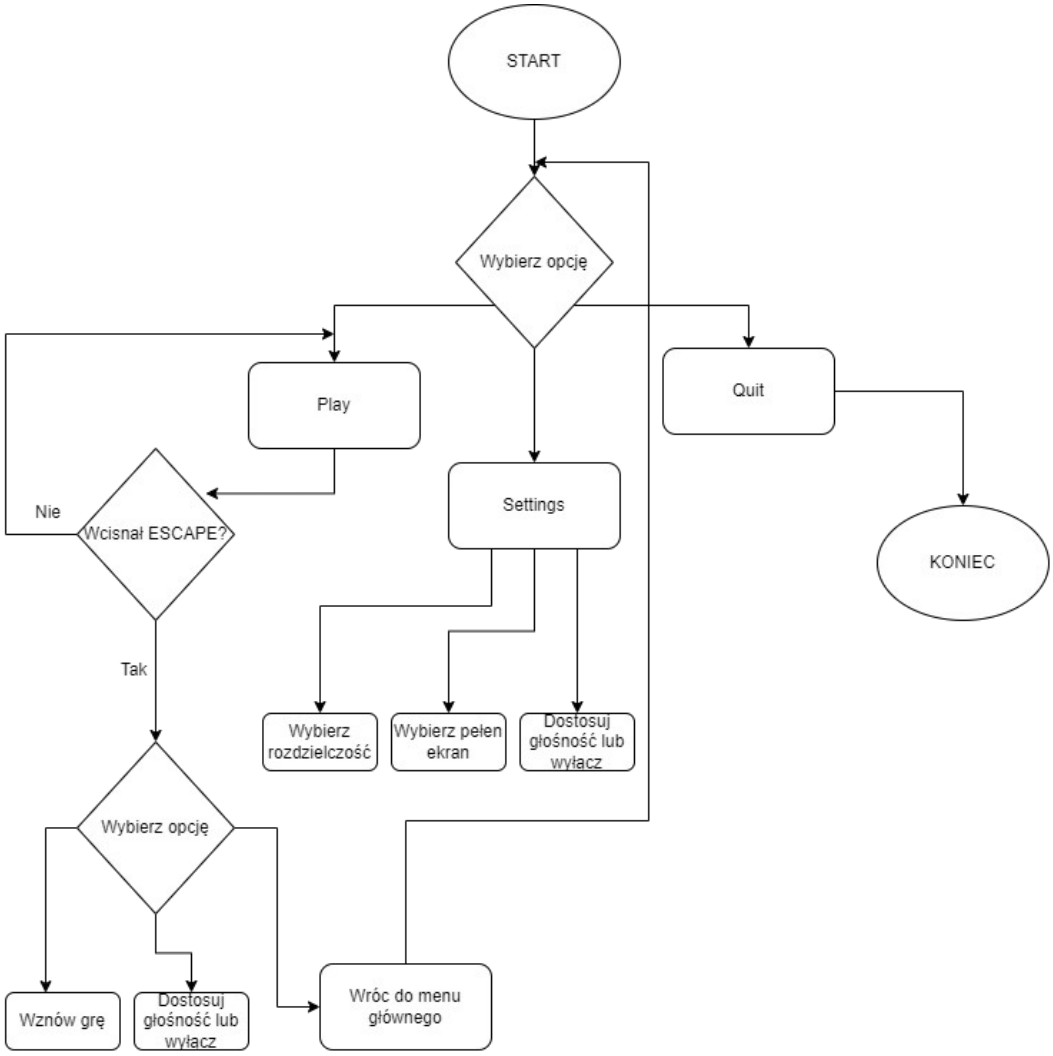
Rys. 1. Wzorzec projektu

W katalogu Controller zostały zawarte 2 klasy. Klasa Bird odpowiada za ruch ptakiem. UIController obsługuje ustawianie oraz także zapisywanie ustawień dźwiękowych. PauseMenu czeka na wciśnięcie przez użytkownika klawisza ESCAPE w celu wyświetlenia menu. Zarządza przyciskami zawartymi w menu.

Katalog Model przechowuje większość klas. Odpowiadają za logikę gry. BackgroundLoop sprawia, że powstaje efekt paralaksy w taki sposób, iż podczas ciągłego ruchu w prawo kamerą, dwa obrazy w tle porusza się wraz z kamerą. Jeden wolniej drugi szybciej. Gdy obrazy wyjdą poza obszar kamery klonowany jest nowy obraz a poprzedni usuwany. GameManager odpowiada za zliczanie punktów oraz waluty, a także rekord w zdobytej ilości punktów. AudioManager realizuje przyjęte przez użytkownika zmiany w ustawieniach dźwięku. Spawnery odpowiadają za tworzenie obiektu oraz przesuwanie go w prawo.

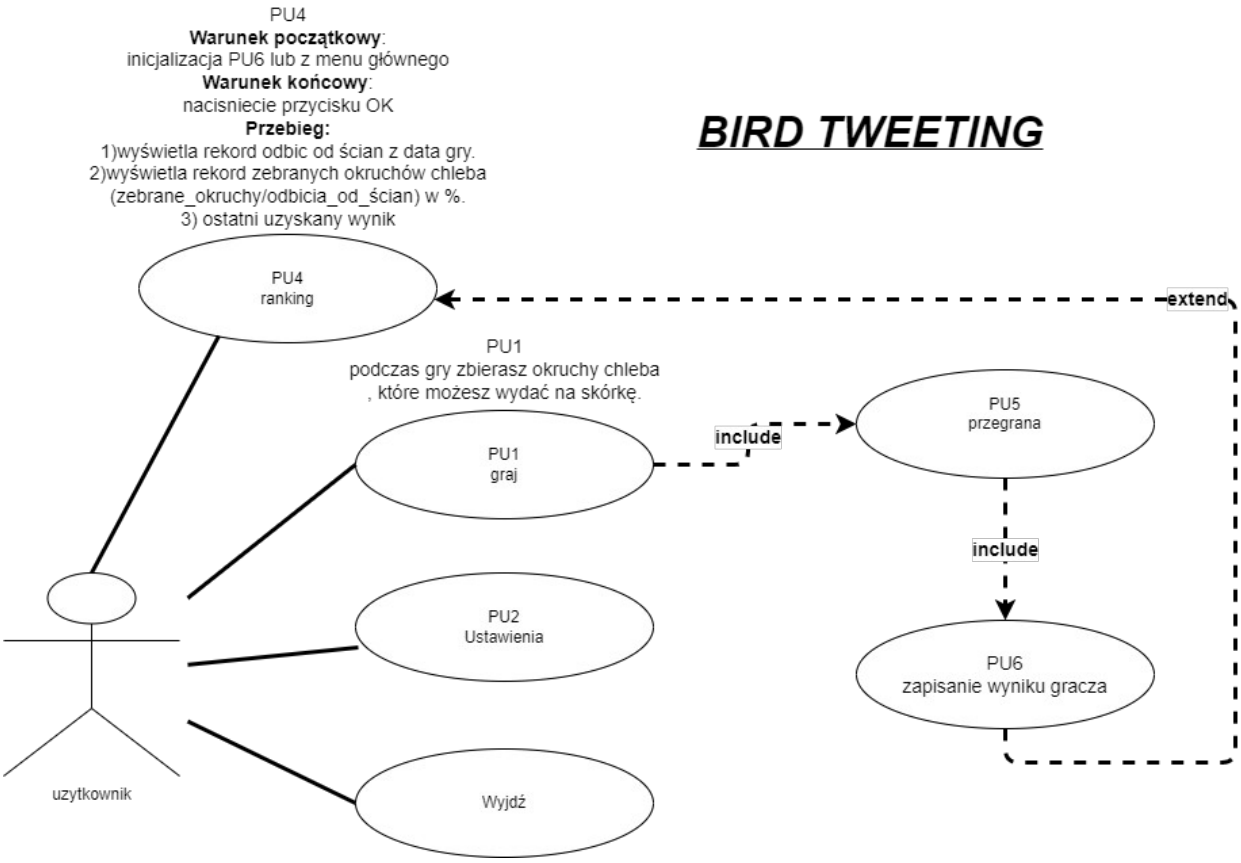
View zawiera część wizualną. Wyświetla przeszkody, chleb i menu. Gdy chleb lub przeszkoda wyjdzie poza kamerę to zostaje usunięta. MainMenu automatycznie pobiera ustawienia monitora użytkownika i na bazie tego pokazuje w opcjach odpowiednie rozdzielczości do wybrania.

Schemat blokowy



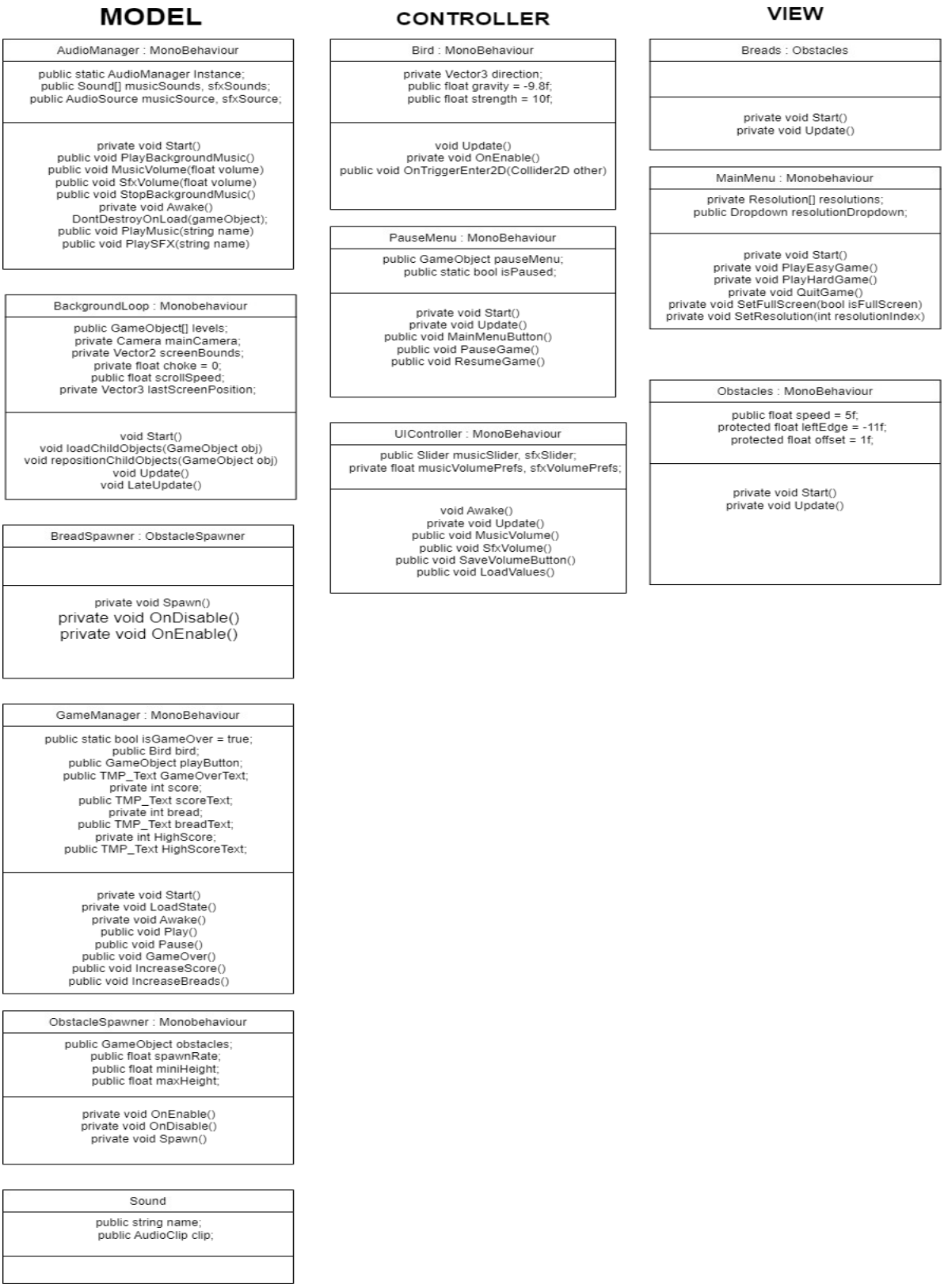
Rys. 2. Schemat blokowy

Use case diagram



Rys. 3.Use case diagram projektu

Diagram klas



Rys. 4 diagram klas

Dziedziczenie

```
public class BreadSpawner : ObstacleSpawner
{
    Event function  pablolambo
    private void OnEnable()
    {
        InvokeRepeating(nameof(Spawn), time: spawnRate, spawnRate);
    }

    Event function  pablolambo
    private void OnDisable()
    {
        CancelInvoke(nameof(Spawn));
    }

    2 usages  pablolambo
    private void Spawn()
    {
        GameObject obstacle = Instantiate(obstacles, transform.position, Quaternion.identity);
        obstacle.transform.position += Vector3.up * Random.Range(miniHeight, maxHeight);
    }
}
```

Rys. 5. Kod klasy BreadSpawner

```
public class ObstacleSpawner : MonoBehaviour
{
    public GameObject obstacles;  Changed in 2+ assets
    public float spawnRate;  Changed in 2+ assets
    public float miniHeight;  Changed in 2+ assets
    public float maxHeight;  Changed in 2+ assets

    Event function  pablolambo
    private void OnEnable()
    {
        InvokeRepeating(nameof(Spawn), time: spawnRate, spawnRate);
    }

    Event function  pablolambo
    private void OnDisable()
    {
        CancelInvoke(nameof(Spawn));
    }

    2 usages  pablolambo
    private void Spawn()
    {
        GameObject obstacle = Instantiate(obstacles, transform.position, Quaternion.identity);
        obstacle.transform.position += Vector3.up * Random.Range(miniHeight, maxHeight);
    }
}
```

Rys. 6. Kod klasy ObstacleSpawner

Klasa BreadSpawner dziedziczy po ObstacleSpawner ponieważ obie klasy używają tych samych pól składowych, lecz różnych wartości, które zmieniam bezpośrednio w Unity.

Fragmenty kodu

```
public void IncreaseScore()
{
    score++;
    scoreText.text = score.ToString();
    if(score > PlayerPrefs.GetInt( key: "HighScore", defaultValue: 0))
    {
        PlayerPrefs.SetInt("HighScore",score);
        HighScoreText.text = score.ToString();
    }
}
```

Rys. 7. fragment kodu klasy GameManager

Metoda IncreaseScore() aktualizuje zmienną score, czyli wynik wyświetlany bezpośrednio w grze. Sprawdza czy użytkownik pobił swój dotychczasowy rekord, jeśli tak to go ustawia.

```
void Update()
{
    if(!PauseMenu.isPaused)
    {
        if (Input.GetMouseButtonDown(0) || Input.GetKeyDown(KeyCode.Space))
        {
            direction = Vector3.up * strength;
            AudioManager.Instance.PlaySFX( name: "Jump");
        }

        if (Input.touchCount > 0)
        {
            Touch touch = Input.GetTouch( index: 0);
            if (touch.phase == TouchPhase.Began)
            {
                direction = Vector3.up * strength;
                AudioManager.Instance.PlaySFX( name: "Jump");
            }
        }

        direction.y += gravity * Time.deltaTime;
        transform.position += direction * Time.deltaTime;
    }
}
```

Rys. 8. Fragment kodu z klasy Bird

metoda Update() jest wzywana co klatkę. Sprawdza czy użytkownik wcisnął LPM, spację albo dotknął ekran i wtedy uaktualnia jego pozycję na ekranie oraz wydaje dźwięk. Na ptaka działa grawitacja, która powoduje, że ciągle spada.

Podsumowanie

Projekt ma potencjał na przyszły rozwój. Na ten moment użytkownik zbiera chleb, ale nie może go wydać, co może być przydatne w przyszłości na zakupy nowej skórki do ptaka.

Link do gry (strona internetowa): https://game-42e45.web.app/

Link do repozytorium na GitHub:
https://github.com/pablolambo/Bird-Tweeting-Game/releases/tag/Game