



# Bird Tweeting

16.01.2023

---

Paweł Frankowski

U-19537

3 semestr

## Wstęp

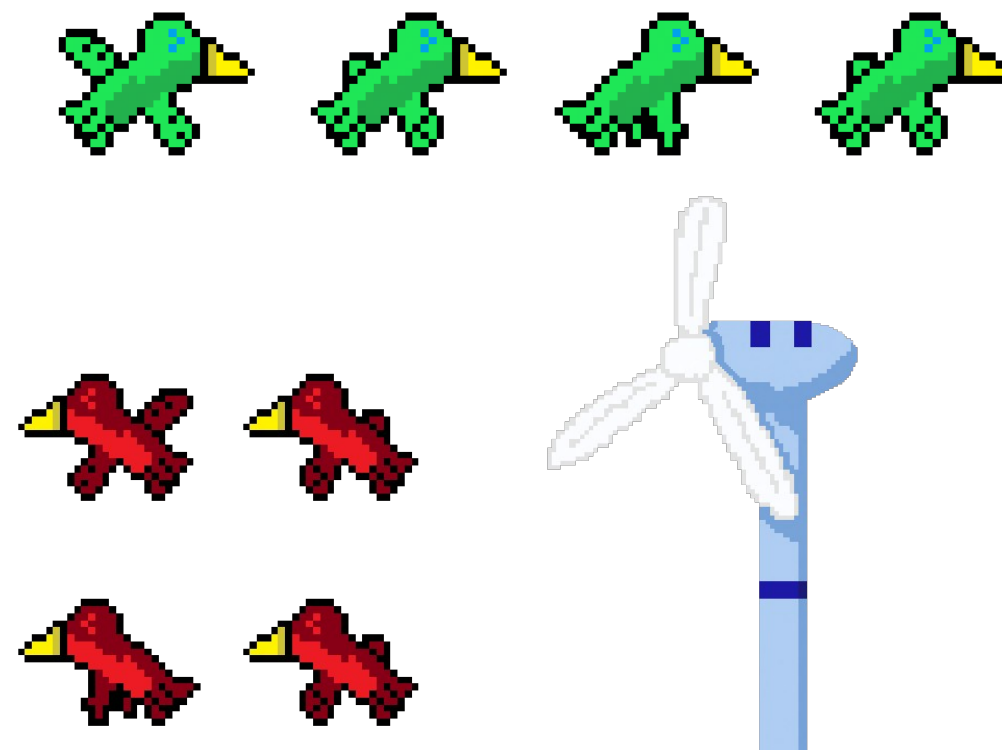
Gra opiera się na zdobywaniu punktów poprzez przelatywanie ptakiem pomiędzy przeszkodami.

## Architektura

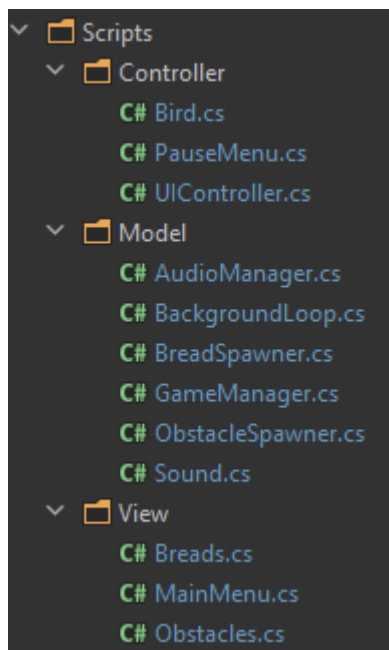
Cała gra opiera się na silniku Unity. Kod źródłowy napisany w 100% w C#.

## 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



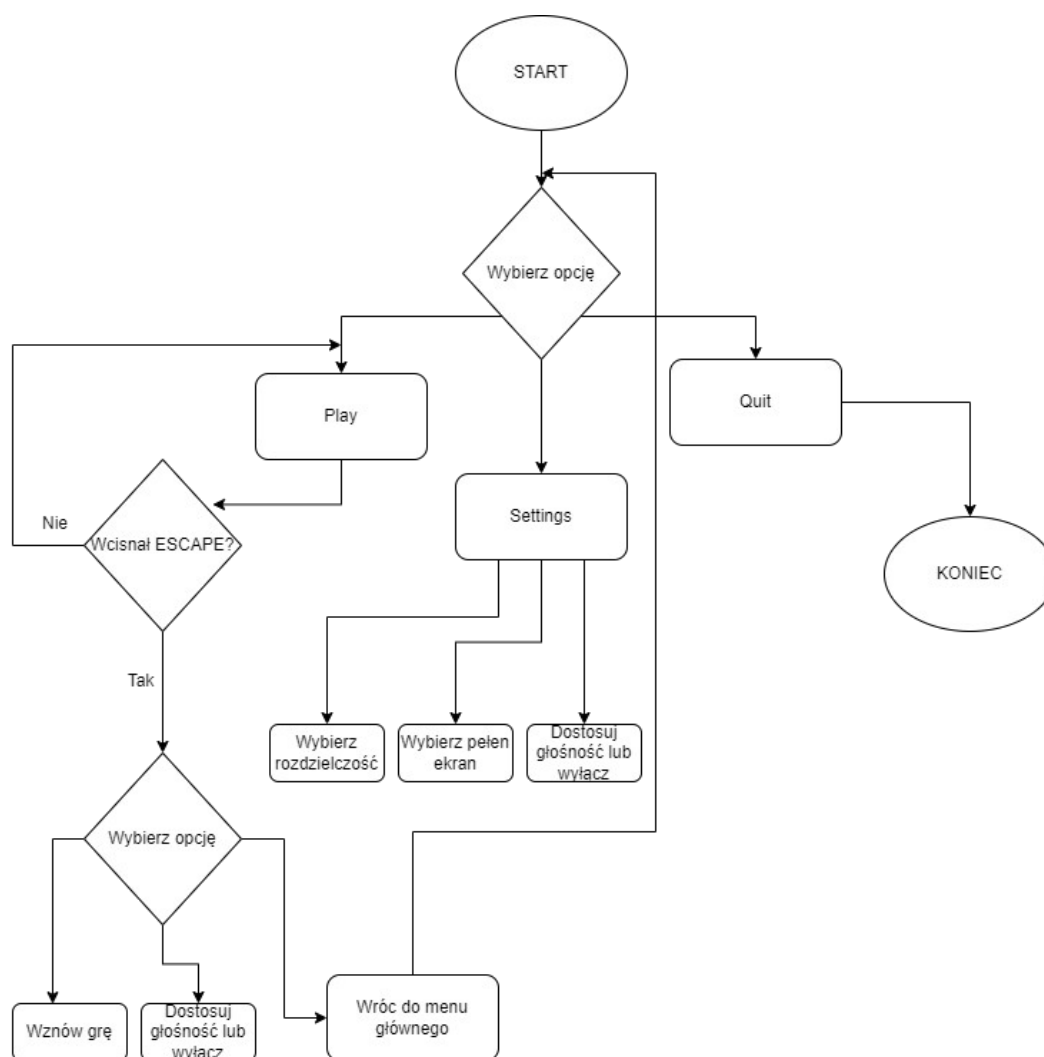
Rys. 1. Wzorzec projektu

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. Spawnerzy odpowiadają za tworzenie obiektu oraz przesuwanie go w prawo.

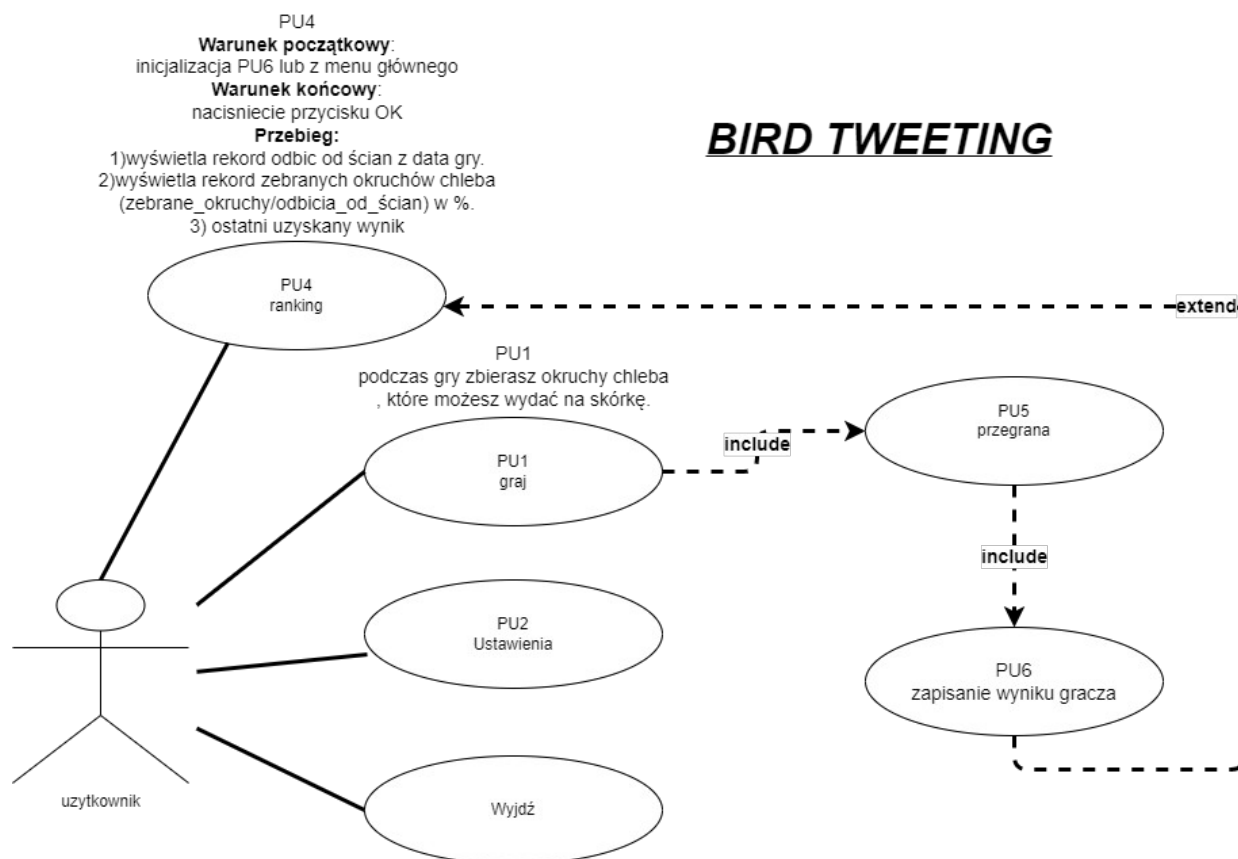
View zawiera część wizualną. Wyświetla przeszkody, chleb i menu. Gdy chleb lub przeszkoda wyjdzie poza kamerę to zostaje usunięta. MainMenu automatycznie pobiera ustawienia monitora użytkownika i na bazie tego pokazuje w opcjach odpowiednie rozdzielczości do wybrania.

## Schemat blokowy



Rys. 2. Schemat blokowy

## Use case diagram



Rys. 3. Use case diagram projektu

# Diagram klas

## MODEL

<b>AudioManager : MonoBehaviour</b> public static AudioManager Instance; public Sound[] musicSounds, sfxSounds; public AudioSource musicSource, sfxSource;  private void Start() public void PlayBackgroundMusic() public void MusicVolume(float volume) public void SfxVolume(float volume) public void StopBackgroundMusic() private void Awake() DontDestroyOnLoad(gameObject); public void PlayMusic(string name) public void PlaySFX(string name)
<b>BackgroundLoop : MonoBehaviour</b> public GameObject[] levels; private Camera mainCamera; private Vector2 screenBounds; private float choke = 0; public float scrollSpeed; private Vector3 lastScreenPosition;  void Start() void loadChildObjects(GameObject obj) void repositionChildObjects(GameObject obj) void Update() void LateUpdate()
<b>BreadSpawner : ObstacleSpawner</b>  private void Spawn() private void OnDisable() private void OnEnable()
<b>GameManager : MonoBehaviour</b> public static bool isGameOver = true; public Bird bird; public GameObject playButton; public TMP_Text GameOverText; private int score; public TMP_Text scoreText; private int bread; public TMP_Text breadText; private int HighScore; public TMP_Text HighScoreText;  private void Start() private void LoadState() private void Awake() public void Play() public void Pause() public void GameOver() public void IncreaseScore() public void IncreaseBreads()
<b>ObstacleSpawner : MonoBehaviour</b> public GameObject obstacles; public float spawnRate; public float miniHeight; public float maxHeight;  private void OnEnable() private void OnDisable() private void Spawn()
<b>Sound</b> public string name; public AudioClip clip;

## CONTROLLER

<b>Bird : MonoBehaviour</b> private Vector3 direction; public float gravity = -9.8f; public float strength = 10f;  void Update() private void OnEnable() public void OnTriggerEnter2D(Collider2D other)
<b>PauseMenu : MonoBehaviour</b> public GameObject pauseMenu; public static bool isPaused;  private void Start() private void Update() public void MainMenuButton() public void PauseGame() public void ResumeGame()
<b>UIController : MonoBehaviour</b> public Slider musicSlider, sfxSlider; private float musicVolumePrefs, sfxVolumePrefs;  void Awake() private void Update() public void MusicVolume() public void SfxVolume() public void SaveVolumeButton() public void LoadValues()

## VIEW

<b>Breads : Obstacles</b>  private void Start() private void Update()
<b>MainMenu : MonoBehaviour</b> private Resolution[] resolutions; public Dropdown resolutionDropdown;  private void Start() private void PlayEasyGame() private void PlayHardGame() private void QuitGame() private void SetFullScreen(bool isFullScreen) private void SetResolution(int resolutionIndex)
<b>Obstacles : MonoBehaviour</b> public float speed = 5f; protected float leftEdge = -11f; protected float offset = 1f;  private void Start() private void Update()

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>