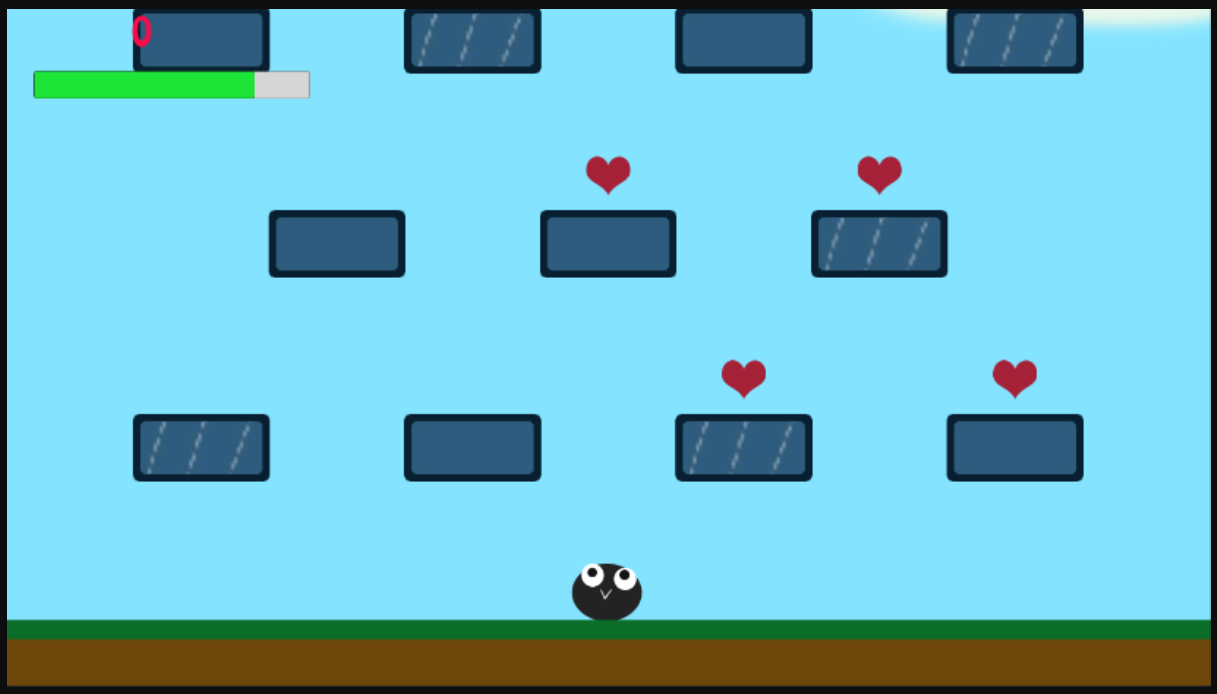
MULTIPLATFORM DEVELOPMENT

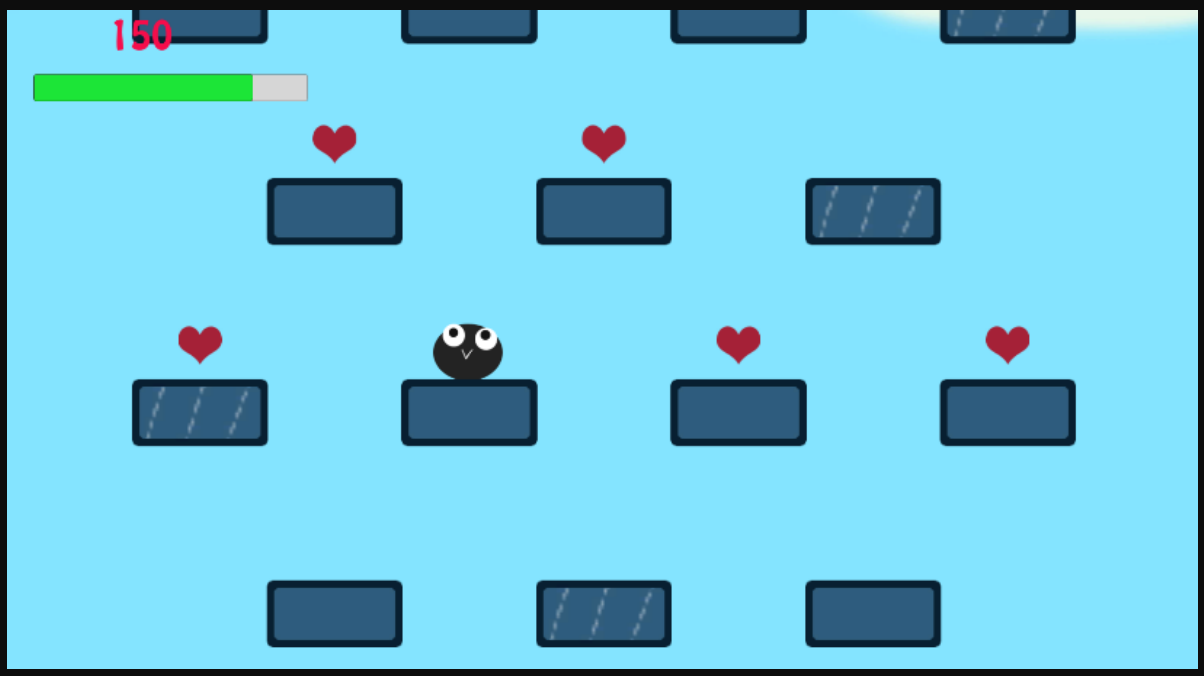
Verslag Lisa Perelaer – Lucky Jump



Concept

Het concept is simpel; het is een namaak van het spelletje Happy Hop (Android) op een versimpelde manier, namelijk zonder procedureel gegenereerd level.
De nadruk ligt hierbij op het simpel houden van het spel zodat eventuele complicaties beter kunnen worden opgelost (en niet dat ik aan het eind vele kleine fouten zit op te lossen).

Het doel is een zo hoog mogelijke scoren halen zonder af te gaan.
De hoogte zelf geeft de score, maar ook het pakken van de hartjes zorgt voor extra punten.
De hartjes zorgen ook voor extra tijd; Je hebt maximaal 5 seconden om bij het volgende hartje te komen.



Platform specifieke code

De Input Class is met behulp van een Abstract Factory platform specifiek. Hier de code:

Base Input (Abstract Factory)

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

public abstract class BaseInput : MonoBehaviour {
    private Vector2 leftVec = new Vector3(-2f, 3f);
    private Vector2 rightVec = new Vector3(2f, 3f);

    public static BaseInput GetPlatform() {
        switch(Application.platform) {
#if !DISABLE_SYSTEM
            case RuntimePlatform.WindowsPlayer:
                return new PCInputClass();
            case RuntimePlatform.Android:
                return new AndroidInputClass();
#endif
            default:
                return new PCInputClass();
        }
    }

    public void Update() {
        Move();
    }

    public virtual float GetInput() {
        return 0f;
    }

    public virtual void Move() {
        float input = GetInput();
        if (input < 0f) {
            transform.Translate(leftVec);
        } else if (input > 0f) {
            transform.Translate(rightVec);
        }
    }
}
```

Hiervan overerven de volgende 2 classes:

PC- Input Class

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

public class PCInputClass : BaseInput {
    public override float GetInput() {
        if (Input.GetButtonDown("Horizontal")) {
            return Input.GetAxisRaw("Horizontal");
        }
        return 0f;
    }

}
```

Android Input Class

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

public class AndroidInputClass : BaseInput {
    public override float GetInput() {
        // check where finger touches
        if (Input.touchCount == 1) {
            Touch touch = Input.GetTouch(0);
            if (Input.GetMouseButtonDown(0)) {
                if (touch.position.x < (Screen.width / 2)) {
                    // if on the left side of the screen, return -1
                    return -1f;
                } else {
                    // if on the right side of the screen, return 1
                    return 1f;
                }
            }
        }
        // else, return 0.
        return 0f;
    }

}
```

Camera Zoom

Deze Class is ook soort van platform specifiek.

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

public class CameraZoom : MonoBehaviour {
    private int screenWidth;
    private int screenHeight;
    Camera cam;


    private void Awake() {
        cam = Camera.main;

        switch (Application.platform) {
            case RuntimePlatform.Android:
                cam.orthographicSize = 12;
                return;
            default:
                cam.orthographicSize = 5;
                return;
        }
    }

    private void Start() {
        switch (Application.platform) {
            case RuntimePlatform.Android:
                Scene currentScene = SceneManager.GetActiveScene();
                if (currentScene.name == "Level") {
                    cam.transform.position = new Vector3(0f, 7f, transform.position.z);
                }

                return;
        }
    }
}
```

Wat ik wel merkte met deze oplossing t.o.v. de
#if
#endif
- is dat het met application.platform niet goed meer werkt als je voor Android test binnen Unity. Dan ziet hij dat namelijk als de default, en er was ook geen optie voor iets als RuntimePlatform.AndroidUnity, of iets dergelijks.
Dus ik snap dat het uiteindelijk iets netter is, maar dan zou ik het eerlijk gezegd alsnog liever op die andere manier willen doen.

Game Code

Generators

Ik heb 2 generators gemaakt die het level 1 keer genereren, namelijk bij het openen van de scene.
1 is de generator van de platforms, die ook bepaald wat voor platform (eentje die kapot gaat of niet) er wordt gegenereerd.

Platform Generator

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

public class PlatformGenerator : MonoBehaviour {
    public GameObject normalPlatformPrefab;
    public GameObject breakPlatformPrefab;
    private GameObject lastSpawnedPrefab;

    public int amountOfPlatforms = 100;
    public float levelWidth = 6f;
    public float yHeight = 3f;

    void Start () {
        GenerateLevel();
    }

    public void GenerateLevel() {
        Vector3 spawnPos = new Vector3(0f, -4.5f);
        for (int i = 0; i < amountOfPlatforms; i++) {
            spawnPos.y += yHeight;
            if (i % 2 != 0) {
                for (int x = -4; x < levelWidth; x += 4) {
                    spawnPos.x = x;
                    GameObject platform = ChoosePlatform();
                    GameObject finalPlat = Instantiate(platform, spawnPos, Quaternion.identity) as GameObject;
                    finalPlat.transform.parent = this.transform;
                }
            } else {
                for (int x = -6; x <= levelWidth; x += 4) {
                    spawnPos.x = x;
                    GameObject platform = ChoosePlatform();
                    GameObject finalPlat = Instantiate(platform, spawnPos, Quaternion.identity) as GameObject;
                    finalPlat.transform.parent = this.transform;
                }
            }
        }
    }
```

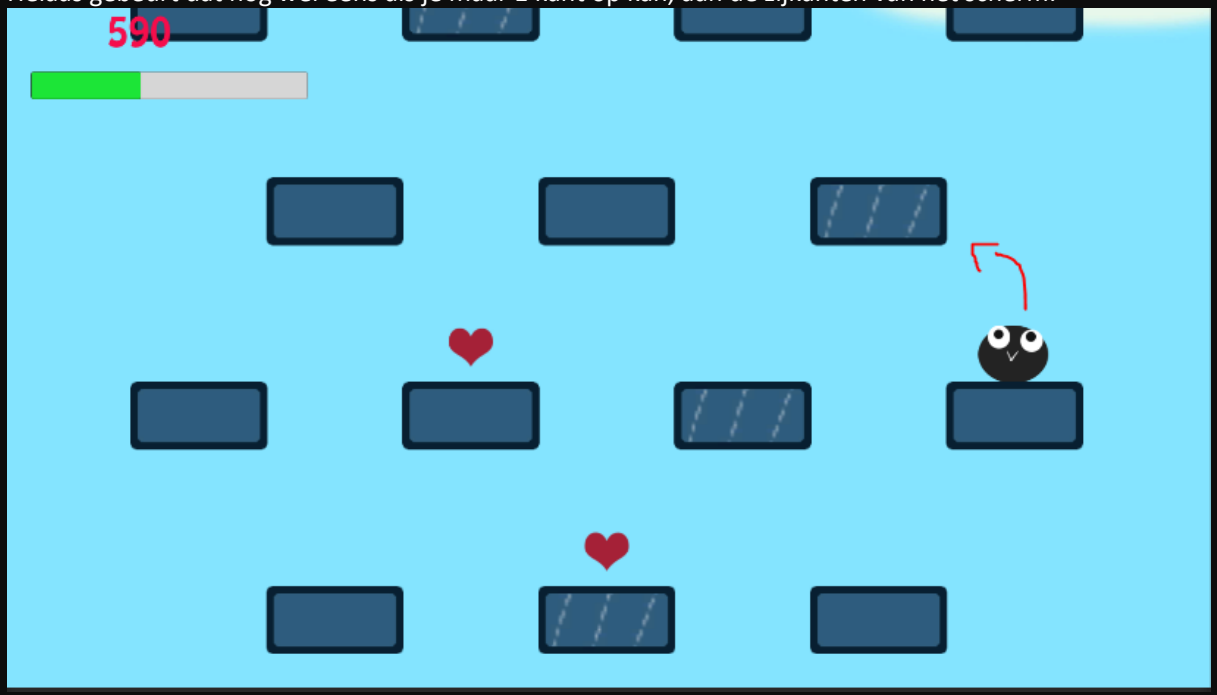
```
public void SetLevelWidth(float width) {
    levelWidth = width;
}

public GameObject ChoosePlatform() {
    GameObject chosenPlatform;

    float value = Random.Range(0f, 1f);
    if (value <= 0.6f) {
        chosenPlatform = normalPlatformPrefab;
    } else {
        chosenPlatform = breakPlatformPrefab;
    }
    if (chosenPlatform == lastSpawnedPrefab) {
        chosenPlatform = normalPlatformPrefab;
    }
    lastSpawnedPrefab = chosenPlatform;
    return chosenPlatform;
}
```

Belangrijk hierbij is dat het genereren in de Start wordt aangeroepen.
In de functie GenerateLevel is het beginpunt aangegegeven (0, -4.5)
Vervolgens wordt er net zolang tot er 100 rijen aan platforms zijn (variabele amountOfPlatforms) gegenereerd. Elke oneven rij moet een stukje inspringen en 1 platform minder hebben.

Vervolgens wordt er elke keer gevraagd of het platform een gebroken of hele moet zijn.
De functie ChoosePlatform geeft er 1 terug.
Ik heb het nooit zo gekregen dat je niet kan vastlopen in het spel, maar in ieder geval wel zo dat er nooit 2 kapotte achter elkaar staan (want dan kan je geen kant op).
Helaas gebeurt dat nog wel eens als je maar 1 kant op kan, aan de zijkanten van het scherm:



De kans dat een platform een kapot platform is is verder 40%.

Om de Hierarchy een klein beetje overzichtelijk te houden zet ik de parent aan de GameManager, het object waar dit script op zit.

Heart Generator

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

public class HeartGenerator : MonoBehaviour {
    public GameObject heartPrefab;

    public int amountOfHearts = 90;
    public float levelWidth = 6f;
    public float yHeight = 3f;
    public bool spawn = false;

    // Use this for initialization
    void Start () {
        GenerateHearts();
    }

    public void GenerateHearts() {
        Vector3 spawnPos = new Vector3(0f, -3.5f);
        for (int i = 0; i < amountOfHearts; i++) {
            spawnPos.y += yHeight;
            if (i % 2 != 0) {
                for (int x = -4; x < levelWidth; x += 4) {
                    bool yes = SpawnOrNot();
                    if (yes) {
                        spawnPos.x = x;
                        GameObject hearts = Instantiate(heartPrefab, spawnPos, Quaternion.identity) as GameObject;
                        hearts.transform.parent = this.transform;
                    }

                }
            } else {
                for (int x = -6; x <= levelWidth; x += 4) {
                    bool yes = SpawnOrNot();
                    if (yes) {
                        spawnPos.x = x;
                        GameObject hearts = Instantiate(heartPrefab, spawnPos, Quaternion.identity) as GameObject;
                        hearts.transform.parent = this.transform;
                    }
                }
            }
        }

    public bool SpawnOrNot() {
        return (Random.value > 0.5f);
    }
}
```

Deze is vrijwel hetzelfde als dat van de platforms, alleen wordt er hier alleen gevraagd of er een hartje uberhaupt moet spawnen (er is geen verder verschil in hartjes).
Dit gebeurt in de functie SpawnOrNot en de kans is 50%.
De hartjes hebben ook de GameManager als parent en ze worden 1 keer aan het begin (in de Start) gespawned.

Singletons & Delegates

Ik heb 3 singleton classes aangemaakt, namelijk:

- ScoreManager
- TimeManager
- UIManager

Ze spreken redelijk voor zich, hoop ik althans, want ik weet wel dat ik singletons echt geweldig vind nu ik ze eenmaal een paar keer heb gebruikt (ondanks dat het zo een veelbesproken onderwerp is of het wel zo goed is- aan de andere kant, voor een klein spel als deze kan het neem ik aan weinig kwaad doen).

ScoreManager

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

public class ScoreManager : MonoBehaviour {
    #region Singleton
    public static ScoreManager instance;

    private void Awake() {
        if (instance != null) {
            return;
        }
        instance = this;
    }
    #endregion


    public Text scoreText;
    public int score = 0;


    public void AddToScore(int num) {
        score += num;
        UpdateScore();
    }

    public void UpdateScore() {
        scoreText.text = score.ToString();
    }

    public void ResetScore() {
        score = 0;
    }

}
```

TimeManager

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

public class TimeManager : MonoBehaviour {
    #region Singleton
    public static TimeManager instance;
    public void Awake() {
        instance = this;
    }
    #endregion

    UIManager uiManager;
    public float time = 10f;

    public Slider timeBar;

    public void Start() {
        uiManager = UIManager.instance;
        timeBar.maxValue = time;
    }

    public void Update() {
        if (time > 5f) {
            time = 5f;
        }
        time -= Time.deltaTime;
        timeBar.value = time;
        if (time <= 0) {
            uiManager.onGameOverCallback.Invoke();
        }
    }

    public void AddTime(int num) {
        time += (float)num;
    }

}
```

Zorgt ervoor dat de timer werkt en wordt geupdate.

UIManager

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

public class UIManager : MonoBehaviour {

    #region Singleton
    public static UIManager instance;

    private void Awake() {
        if (instance != null) {
            return;
        }
        instance = this;
    }
    #endregion

    // delegates
    public delegate void OnGameState();
    public OnGameState onGameOverCallback;
    public OnGameState onWinCallback;

    // Panels
    public GameObject GameOverPanel;
    public GameObject winPanel;


    void Start () {
        onGameOverCallback += ShowGameOverUI;
        onWinCallback += ShowWinUI;


    }

    public void ShowWinUI() {
        winPanel.SetActive(true);
    }

    public void ShowGameOverUI() {
        GameOverPanel.SetActive(true);
    }
}
```

Zet de WinPanel aan als de Win situatie wordt ge-invoked, en zet de GameOverPanel aan als de Game over situatie wordt ge-invoked.

Puntensysteem

Ik heb 1 script aangemaakt die de punten regelt en de tijd.

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

public class Points : MonoBehaviour {
    ScoreManager scoreManager;
    public int score;
    public bool destroyable;

    TimeManager timeManager;
    public int plusTime;

    private void Start() {
        scoreManager = ScoreManager.instance;
        timeManager = TimeManager.instance;
    }

    private void OnTriggerEnter2D(Collider2D collision) {
        scoreManager.AddToScore(score);
        if (plusTime != 0) {
            timeManager.AddTime(plusTime);
        }
        if (destroyable) {
            Destroy(gameObject);
        }

    }
}
```

Je kan hier aangeven of je punten wil toevoegen (en in de inspector hoeveel, per prefab waarop het script staat).

Ook kan je zeggen of het object kapot moet als je door de trigger heen bent gegaan. Als je destroyable aanvinkt, wordt het object gedestroyed als je door de trigger gaat. Dit is bijvoorbeeld voor de hartjes wel zo, maar niet voor de punten die je krijgt als je op een platform springt.