Cloud Hopper

A Flappy-Bird style platformer

Florens Shosho & Michael Gorman

What platform did we use?

To create this project we used the Unity-2D framework, which allowed us to produce our app for multiple OS platforms, namely Android, and iOS.

The Unity engine is used to create 2D, 3D, VR and AR applications

Unity provided many useful tools to make app integration simply and easy

For instance, one click of a button allowed us to change the build platform from iOS to Android

Our project was written in C#, a language we had to learn and get familiar with

Database?

```
[RequireComponent(typeof(Text))]
public class TotalScore : MonoBehaviour
{

    Text totalScore;

    void OnEnable()
    {
        totalScore = GetComponent<Text>();
        totalScore.text = "Total Score: " + PlayerPrefs.GetInt("TotalScore").ToString();
    }
}
```

No database was used for this project

Unity Engine API comes with a way to save and store player data during runtime, therefore we did not see a need for a database like Firebase

PlayerPrefs allows us to store all of the needed player data (highscores, trophy system, etc) in JSON for easy readability and accessibility

https://docs.unity3d.com/ScriptReference/PlayerPrefs.html

Prefabs

We learned the utility of Unity's Prefabs, which is a system that allowed us to store our GameObjects complete with all of its components/property values

This was particularly useful when we wanted to reuse our CloudSmall and Cloud Big objects to appear in multiple places in our screen

The Prefab system helped keep our object copies in sync with each other, allowing for an easier transition in building our Paralaxer

“WhAt tHe HeCk iS A pAraLaXeR?”

To be more specific, our Parallax Scrolling Controller utilizes a technique where background images move past the camera more slowly than foreground images

This creates an illusion of depth in 2D scenes and can add a sense of immersion in the virtual experience

This is a technique that grew out from the use of traditional animation since the 1930s

```
    void SpawnImmediate() {
        Transform t = GetPoolObject();
        if (t == null) return;  // if true, pool size is too small
        Vector3 pos = Vector3.zero;
        pos.x = immediateSpawnPos.x;
        pos.y = Random.Range(spawnRange.min, spawnRange.max);   // pos.y is a
random value
        t.position = pos;   // set position of our pool object
        Spawn();
    }

    void Shift() {
        for (int i = 0; i < poolObjectsArray.Length; i++) {
            //  move poolObject to the left at a constant pace
            poolObjectsArray[i].transform.localPosition += -Vector3.right *
shiftSpeed * Time.deltaTime;
            //  Check if its disposed
            CheckDisposedObject(poolObjectsArray[i]);
        }
    }

    //  Check to see if parallax object is off-screen, if it is, set inUse to false using
Dispose()
    void CheckDisposedObject(PoolObject poolObject) {
        //  if our poolObject is on the left side of the camera
        if (poolObject.transform.position.x < -defaultSpawnPos.x) {
            //  set as unused
            poolObject.Dispose();
            // set position to someplace off-screen to the right
            poolObject.transform.position = Vector3.one * 1000;
        }
    }
```

Tap Controller

The Tap Controller is in charge of the players movement, causing the player to fly higher when phone screen is tapped

In addition to the movement, it is in charge of setting the players position back to default after a gameover

Finally, it is in charge of signaling the onPlayerDied() & onPlayerScored() in order to do appropriate updates to PlayerPrefs

```
    // Update is called once per frame
    void Update() {
        if (gameManager.getGameOver()) return;
        if (Input.GetMouseButtonDown(0)) { //  This capturing the tap notification on mobile devices
            rigidBody.velocity = Vector3.zero;  // We don't want to add our new force with the current force
            transform.rotation = forwardRotation;
            rigidBody.AddForce(Vector2.up * tapForce, ForceMode2D.Force);
        }
        transform.rotation = Quaternion.Lerp(transform.rotation, downRotation, tiltness * Time.deltaTime);
    }

    void OnTriggerEnter2D(Collider2D collider) {
        if (collider.gameObject.tag == "ScoreZone") {
            //register score events
            OnPlayerScored();   // event sent to GameManager
            //play sound
            // etc
        }
        if (collider.gameObject.tag == "DeadZone") {
            rigidBody.simulated = false;    //  Freeze player if dead
            // register a dead event
            OnPlayerDied(); // event sent to GameManager
            // play sound
            //etc
        }
    }
}
```

Game Manager

Our Game Manager class handles the player's score during runtime

It is also in charge of the changing of our UI's Page States, allowing the user to switch to the different Play, Main Menu, Trophies screens

Finally, the Game Managers is the part actually handling the Player Scored, Player Death, and Game Over scenarios

```
void OnPlayerDied() {
    gameOver = true;
    int currentHighScore = PlayerPrefs.GetInt("HighScore");  //  Get current high
score
    int totalDeaths = PlayerPrefs.GetInt("TotalDeaths");
    totalDeaths++;
    PlayerPrefs.SetInt("TotalDeaths", totalDeaths);
    if (duringGameScore > currentHighScore) {
        PlayerPrefs.SetInt("HighScore", duringGameScore);
    }

    setPageState(PageState.GameOver);

}

void OnPlayerScored() {
    duringGameScore++;
    int currentTotalScore = PlayerPrefs.GetInt("TotalScore");
    currentTotalScore++;
    PlayerPrefs.SetInt("TotalScore", currentTotalScore);
    PlayerPrefs.SetInt("RunningScore", duringGameScore);
    scoreText.text = duringGameScore.ToString();
}


void setPageState(PageState state) {
    switch (state) {
        case PageState.None:
            startMenu.SetActive(false);
            gameOverMenu.SetActive(false);
            countDownMenu.SetActive(false);
            settingsMenu.SetActive(false);
            trophiesMenu.SetActive(false);
            break;
        case PageState.Start:
            startMenu.SetActive(true);
            gameOverMenu.SetActive(false);
            countDownMenu.SetActive(false);
            settingsMenu.SetActive(false);
            trophiesMenu.SetActive(false);
            break;
        case PageState.GameOver:
            startMenu.SetActive(false);
```

Cloud Hopper Video & GitHub Link

Youtube Demo:

https://www.youtube.com/watch?v=hhZs-ZznxDc&feature=youtu.be

GitHub Link:

https://github.com/floshosho/CloudHopper