CANDY SURFER

CANDY SURFER

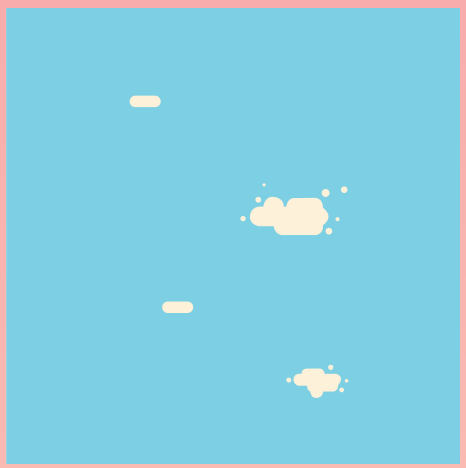
PLAY

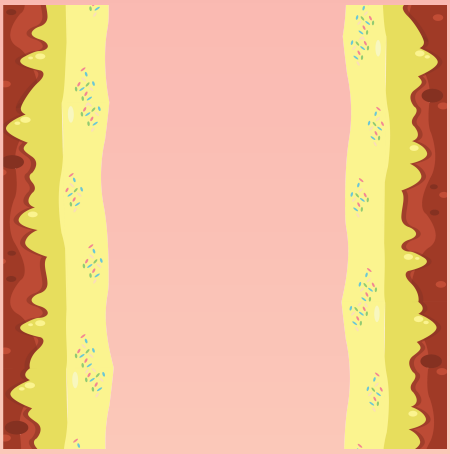
QUIT

GRAPHICAL ASSETS

THIS WAS USED AS THE MAIN CHARACTER OF THE GAME. THIS WAS SET FOR THE PLAYER TO MOVE THE CHARACTER FROM LEFT TO RIGHT.

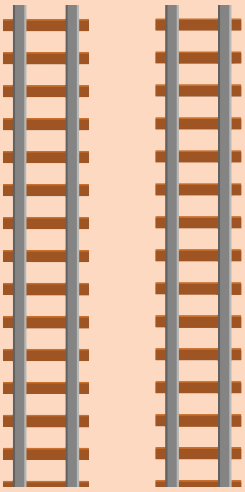
WE GAVE THE BACKGROUND A LIGHT BLUE COLOUR WITH CLOUDS IN ORDER TO GIVE THE APPEARANCE OF AN OUTDOOR SETTING.





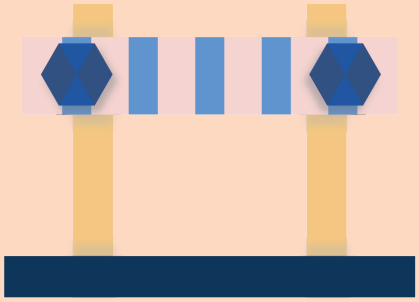
ANOTHER ART ASSET THAT MAKES PART OF THE BACKGROUND. WE TRIED TO MIMIC A PAVEMENT WITH THE CANDY LAND THEME.

BACKGROUND EXTRAS- CONSISTS OF CANDY AND TRAIN. THESE ART ASSETS ENHANCED THE THEME OF THE GAME AND MADE THE BACKGROUND MORE INTERESTING AND VISUALLY APPEALING THE PLAYER.



BACKGROUND EXTRAS- CONSISTS OF CANDY AND TRAIN. THESE ART ASSETS ENHANCED THE THEME OF THE GAME AND MADE THE BACKGROUND MORE INTERESTING AND VISUALLY APPEALING THE PLAYER.

WE CREATED THIS ART ASSET TO ACT AS AN OBSTACLE. WHEN THE PLAYER HITS THE OBSTACLE, THE GAME WILL BE OVER.



WE CREATED THIS ART ASSET TO ACT AS AN OBSTACLE. WHEN THE PLAYER HITS THE OBSTACLE, THE GAME WILL BE OVER.

THE PLAY BUTTON IS USED SO THAT THE PLAYER CAN START PLAYING THE GAME.

PLAY

QUIT

THE QUIT BUTTON IS USED SO THE PLAYER CAN QUIT THE GAME.

SCRIPTS OVERVIEW

Users > christinepacerivans > Downloads > SubwaySurfer > Assets > Scripts > CharacterScript.cs

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

public class CharacterScript : MonoBehaviour
{
    public float moveSpeed = 5f; //this is for the characters speed

    public bool isGrounded = false;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        Jump();

        Vector3 movement = new Vector3(Input.GetAxis("Horizontal"), 0f, 0f);

        transform.position += movement * Time.deltaTime * moveSpeed;
    }

    void Jump(){
        if (Input.GetButtonDown("Jump") && isGrounded == true) {
        gameObject.GetComponent<Rigidbody2D>().AddForce(new Vector2(0f, 10f), ForceMode2D.Impulse);
        }
    }
}

```

CHARACTER SCRIPT- THIS SCRIPT WAS CREATED FOR THE CHARACTER. WE INCLUDED THE MOVEMENT WHICH IS A HORIZONTAL MOVEMENT AND ALSO THE SPEED OF THE CHARACTER WHICH IS 5F. AS A RESULT, THE CHARACTER ONLY MOVES FROM LEFT TO RIGHT BY PRESSING THE LEFT AND RIGHT ARROWS ON THE KEYBOARD WITH SPEED 5 F.

OBSTACLE- THIS SCRIPT WAS. CREATED FOR THE OBSTACLE IN ORDER FOR THE OBSTACLE TO MOVE ON THE DOWNWARDS ON THE Y AXIS WITH A SPEED OF 7F. WE ALSO INCLUDED A CODE ON ORDER FOR THE OBSTACLE TO BE GENERATED IN. A RANDOM POSITION WITH A RANGE OF -1F TO 1.6F.

Users > christinepacerivans > Downloads > SubwaySurfer > Assets > Scripts > Obstacles > Obstacle.cs

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

public class Obstacle : MonoBehaviour
{
    public float speed = 7f;


    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
       transform.Translate(Vector3.down * Time.deltaTime * speed);
       if(transform.position.y < -6f)
       {
            float random_X = Random.Range(-1f , 1.6f);
            transform.position = new Vector3(random_X,8,transform.position.z);
       }
    }
}

```

Users > christinepacerivans > Downloads > SubwaySurfer > Assets > Scripts > Obstacles >

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

public class ObstacleCollisionHandler : MonoBehaviour
{
    // Start is called before the first frame update
    void OnTriggerEnter2D(Collider2D other)
    {
        Debug.Log("The player has entered obstacle mode");
        //reload the scene when the player hits the obstacles
        SceneManager.LoadScene(0);
    }
}

```

OBSTACLE COLLISION HANDLER- THIS SCRIPT WAS CREATED IN ORDER TO LOAD SCENE (END GAME) WHEN THE PLAYER HITS THE OBSTACLE.

BACKGROUND SCRIPT

Users > christinepacerivans > Downloads > SubwaySurfer > Assets > Scripts > Scripts Background > CakeInfiniteScroll.cs

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

public class CakeInfiniteScroll : MonoBehaviour
{
    [SerializeField]
    private float _speed = 2f;
    // Start is called before the first from update

    void Start()
    {

    }

    // Update is called once per frame

    void Update()
    {
        //Vector3.down = Vector3(0,-1,0)
        //Vector3.down (-1) * 1 * _speed(3) = -3
        transform.Translate(Vector3.down * Time.deltaTime * _speed);

        //check the y position of the train tracks if it is smaller than -17
        if(transform.position.y < -17)
        {
            //teleport to 17 on the y axis
            transform.position = new Vector3(0,17,0);
        }

    }
}
```

CAKE INFINITE SCROLL- THIS SCRIPT WAS CREATED IN ORDER FOR THE BACKGROUND TO MOVE -Y WITH A SPEED OF 2F AND TO ALSO BE GENERATED. THE BACKGROUND IS CREATED USING A RECYCLING SYSTEM SO THAT WHEN THE POSITION OF THE CAKE BACKGROUND REACHES -17 ON THE Y AXIS IT DISAPPEARS AND REAPPERS AT THE TOP OF THE SCREEN.

SKY INFINITE SCROLL- THIS SCRIPT IS THE SAME AS THE CAKE INFINITE SCROLL WITH ALSO THE SAME SPEED AND RECYCLING SYSTEM, WHICH RECYCLES WHEN IT REACHES -16 ON THE Y AXIS.

Users > christinepacerivans > Downloads > SubwaySurfer > Assets > Scripts > Scripts Background > SkyInfiniteScroll.cs

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

public class SkyInfiniteScroll : MonoBehaviour
{
    [SerializeField]
    private float _speed = 2f;
    // Start is called before the first from update

    void Start()
    {

    }

    // Update is called once per frame

    void Update()
    {
        //Vector3.down = Vector3(0,-1,0)
        //Vector3.down (-1) * 1 * _speed(3) = -3
        transform.Translate(Vector3.down * Time.deltaTime * _speed);

        //check the y position of the train tracks if it is smaller than -16
        if(transform.position.y < -16)
        {
            //teleport to 16 on the y axis
            transform.position = new Vector3(0,16,0);
        }

    }
}
```

Users > christinepacerivans > Downloads > SubwaySurfer > Assets > Scripts > Scripts Background > TrainTracksInfiniteScroll.cs

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

public class TrainTracksInfiniteScroll : MonoBehaviour
{
    [SerializeField]
    private float _speed = 2f;
    // Start is called before the first from update

    void Start()
    {

    }

    // Update is called once per frame

    void Update()
    {
        //Vector3.down = Vector3(0,-1,0)
        //Vector3.down (-1) * 1 * _speed(3) = -3
        transform.Translate(Vector3.down * Time.deltaTime * _speed);

        //check the y position of the train tracks if it is smaller than -18
        if(transform.position.y < -18)
        {
            //teleport to 18 on the y axis
            transform.position = new Vector3(0,18,0);
        }

    }
}
```

TRAIN TRACKS INFINITE SCROLL- THIS SCRIPT WAS ALSO CREATED LIKE THE PREVIOUS TWO BACKGROUND SCROLLS. INCLUDING ALSO THE RECYCLING SYSTEM WHICH THIS TIME WHEN THE BACKGROUND REACHES -18 ON TH Y AXIS IT RESTORES TO ITS ORIGINAL POSITION AGAIN.

SCORE KEEPER- THE SCORE SCRIPT WAS CREATED IN ORDER TO KEEP TRACK OF THE PLAYER S SCORE. THE SCORE WILL INCREASE WHENEVER THE PLAYER HITS THE CANDIES STARTING ALWAYS FROM 0.

Users > christinepacerivans > Downloads > SubwaySurfer > Assets > Scripts > Scores > ScoreKeeper.cs

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

public class ScoreKeeper : MonoBehaviour
{
    [SerializeField]

    private int _currentScore = 0;
    // Start is called before the first frame update

    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }

    public void IncrementScore()
    {
        //_currentScore + 1
        //_currentScore += 1
        _currentScore ++;
        Text scoreText = GetComponent<Text>();
        scoreText.text = "Score: " + _currentScore.ToString();
    }
}
```

Users > christinepacerivans > Downloads > SubwaySurfer > Assets > Scripts > Scores > CandyScore.cs

```
public class CandyScore : MonoBehaviour
{
    [SerializeField]

    private float _speed = 1f;
    // Start is called before the first frame update
    void Start()
    {
        Reset();
    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.down * Time.deltaTime * _speed);
        if(transform.position.y < -4.9)
        {
            Reset();
        }
    }

    void Reset()
        {
            float randomHeight = Random.Range(-5f,7f);
            transform.position = new Vector3(transform.position.x,randomHeight,transform.position.z);
        }

    void OnTriggerEnter2D(Collider2D other)
        {
            //we need to check if we collided with the player
            if(other.tag == "Player")
            {
                //create a reference to the scorekeeper
                ScoreKeeper scoreKeeper = GameObject.FindObjectOfType<ScoreKeeper>();
                //we need to check if the script was found
                if(scoreKeeper != null)
                {
                    //IncrementScore
                    scoreKeeper.IncrementScore();
                }
                //reset()
                Reset();
            }
        }
```

CANDY SCORE- IN THIS SCRIPT WE INCLUDED THE SPEED OF THE CANDY AND MOVEMENT ON THE Y-AXIS. THE CANDY IS GENERATED THROUGHOUT THE GAME ON THE Y AXIS WITH A RANDOM POSITION THAT HAS A RANGE FROM -5F TO 7F. WE ALSO INCLUDED A CODE THAT IS USED WHEN THE PLAYER COLLIDES WITH THE CANDY THE SCORE INCREASES.

CHERRY SCORE- THIS SCRIPT IS THE SAME AS THE CANDY SCRIPT, IT GENERATES ALSO THE CHERRIES IN A RANDOM RANGE BETWEEN -5F TO 7F AND SCORE IS INCREASED WHEN COLLIDING WITH THE CHERRY.

Users > christinepacerivans > Downloads > SubwaySurfer > Assets > Scripts > Scores > CherryScore.cs

```
public class CherryScore : MonoBehaviour
{
    [SerializeField]

    private float _speed = 1f;
    // Start is called before the first frame update
    void Start()
    {
        Reset();
    }

    // Update is called once per frame
    void Update()
    {
        transform.Translate(Vector3.down * Time.deltaTime * _speed);
        if(transform.position.y < -4.9)
        {
            Reset();
        }
    }

    void Reset()
        {
            float randomHeight = Random.Range(-5f,7f);
            transform.position = new Vector3(transform.position.x,randomHeight,transform.position.z);
        }

    void OnTriggerEnter2D(Collider2D other)
        {
            //we need to check if we collided with the player
            if(other.tag == "Player")
            {
                //create a reference to the scorekeeper
                ScoreKeeper scoreKeeper = GameObject.FindObjectOfType<ScoreKeeper>();
                //we need to check if the script was found
                if(scoreKeeper != null)
                {
                    //IncrementScore
                    scoreKeeper.IncrementScore();
                }
                //reset()
                Reset();
            }
        }
```

PROCESS

WE FIRST THOUGHT ABOUT HYPER-CASUAL GAMES THAT WERE SUITABLE TO BE REPLICATED. WE BOTH AGREED ON SUBWAY SURFER. AFTER THAT, WE STARTED JOTTING DOWN IDEAS ON MAKING THE GAME DIFFERENT FROM THE ORIGINAL ONE. WE DECIDED TO STICK WITH A CANDY LAND THEMED GAME.

FROM THAT STEP, WE DESIGNED THE GAME ON PAPER, AND BOTH OF US STARTED WORKING ON ART ASSETS. WHEN THE ART ASSETS WERE DONE, WE JUMPED INTO UNITY AND BEGAN WITH THE BACKGROUND. WE WANTED TO ACHIEVE AN INFINITE BACKGROUND. WE HAD SOME FAILS WHEN WORKING ON THE BACKGROUND, BUT WITH OUR RESEARCH AND OUR LECTURER GUIDING US, WE MANAGED TO HAVE AN ENDLESS BACKGROUND USING A RECYCLING SYSTEM IN ORDER FOR THE ASSET TO REACH A DESIRED POSITION AND RESTORE BACK TO ITS ORIGINAL PLACE.

AFTER THE BACKGROUND WAS DONE, WE MOVED ON THE CHARACTER. AS FOR THE CHARACTER, WE DID NOT COME ACROSS ANY DIFFICULTIES.THE ONLY THING THAT WE HAD WAS THE WHEN THE CHARACTER REACHES THE ALLOCATED SIDES OF THE SCREEN IT IS NOT WORKING HOW WE THOUGHT BUT WE WERE ADVISED THAT IT WAS BECAUSE WE USED A DIFFERENT SCRIPT.

AFTER THAT, WE MOVED ON TO THE OBSTACLE. THE LECTURER GAVE US SOME HINTS ON WHAT CODING WE NEEDED TO USE. WE DECIDED THAT THE GAME NEEDS TO HAVE A SCORE WE ALSO ADDED SWEETS TO THE GAME. WHEN THE PLAYER COLLIDES WITH THE SWEETS, THE SCORE INCREASES BY 1.

WHEN THAT WAS DONE, WE CREATED THE MAIN MENU WITH THE OPTIONS OF PLAY AND QUIT AND INCLUDED ALSO A TITLE. WE HAD SOME ISSUES WITH THE PLAY BUTTON AS WHEN THE GAME OPENED THE GAME STARTED, AND THE SCORE WAS SHOWING ON THE MAIN MENU. AS A RESULT WE FIXED SOME CODING AND GOT IT RIGHT. IMPORTED AN AUDIO AND THE GAME WAS DONE.

TESTING

FUNCTIONALITY	EXPECTED OUTPUT	ACTUAL OUTPUT	
PRESSING THE LEFT ARROW KEY MOVES CHARACTER TO THE LEFT	CHARACTER MOVES TO THE LEFT	CHARACTER MOVES TO THE LEFT	
PRESSING THE RIGHT ARROW KEY MOVES CHARACTER TO THE RIGHT	CHARACTER MOVES TO THE RIGHT	CHARACTER MOVES TO THE RIGHT	
COLLIDING WITH SWEET ADDS +1 TO SCORE	SCORE + 1	SCORE + 1	
PLAYER HITS OBSTACLE	GAME RESETS	GAME RESETS	
PLAYER PRESSES QUIT BUTTON	APPLICATION CLOSES	APPLICATION DOES NOT CLOSE	
PLAYER PRESSES PLAY BUTTON	GAME STARTS	GAME STARTS	
GAME STARTS, SCORE STARTS	SCORE INCREASES EVERY SECOND	SCORE IS NOT INCREASING EVERY SECOND	
PLAYER HITS SIDEWALLS	PLAYER COLLIDES SMOOTHLY	PLAYER IS NOT COLLIDING SMOOTHLY	
CANDY SPAWNING	SPAWNING IN RANDOM POSITIONS	RANDOM SPAWNING IS NOT GOING AS PLANNED	