PROGRESS REPORT

MICROGAME #2: Breakout

Andrew Adame

ID: 007516100

LEGEND: COMPLETED – UNFINISHED – WIP – FIX - FIXED

GITHUB LINK: https://github.com/andrewadame/UnityProjectsCSE-4410/tree/master/BreakoutProject

1. Create new project Breakout
2. Create folders containing important assets (Scripts, prefabs, etc)
3. Create basic Breakout Game
    a. Design Level
        i. Design Two Levels
            1. Scene Name: Lvl1
                a. Basic Breakout level with 23 Bricks
            2. Scene Name: Lvl2
                a. Different pattern, more complex
                b. Heart-shape pattern coincidentally made on Valentine's Day :3
    b. Paddles
        i. Paddle Script
            1. Called 'PlyrController'
        ii. Player-Controlled
        iii. Preferred Physics
            1. Paddle stops/starts immediately based on input
    c. Ball
        i. Ball Script
            1. Called 'BallController'
        ii. Movement
        iii. Physics
            1. ERRORS/PROBLEMS
                a. Ball stagnates (bounces back and forth in same spot)
                    ~~i. Temporary solution; add reset button?~~
                    ii. Add random direction command in BallController based on wall impact?
                    iii. Give player slight controller over ball bounce direction?

        b. Ball clips through objects at certain high speeds
        c. Ball sometimes slow down to near stop

d. Gameplay
    i. Game Script
        1. Called ‘GameController’
    ii. Start
        1. Ball spawns, launched towards player
    iii. Bounces between bricks and players
        1. Bricks destroy upon ball impact
    iv. Fail Goal
        1. Life lost upon paddle missing ball
    v. Level Completion
        1. Move to next level upon destroying all bricks of a level.

e. UI
    i. Brick Amount Tracking
        1. Tracks amount of Bricks left
        2. Goes down upon destruction of Bricks
    ii. Live Tracking
        1. Tracks player lives before game over
        2. Goes down upon Ball making it into Fail Goal
    iii. Game Over
        1. Game Over screen with option to restart upon input of any key

f. **EXTRA**
    i. Debugging/Error Solutions
        1. Add random direction command in BallController based on wall impact
        2. Give player slight controller over ball bounce direction
        3. Create controlled speed multiplier
    ii. Audio
        1. Added sounds to ball impacting player, bricks, and goal.
            a. Sounds used are from original Atari “Pong”
    iii. More Levels
    iv. Choose amount of lives to end game?
    v. Time option?
    vi. Restart level option?
    vii. Quit game option?
    viii. Score tracking?
    ix. Ball modifiers
    x. Brick modifiers
        1. Multi-Ball Brick?

SCRIPTS

1. BallController

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

public class BallController : MonoBehaviour
{

    Rigidbody2D ballRigidbody;
    public float speed;
    public float randomUp;

    Vector3 startPosition;

    GameController cont;

    //Audio
    private AudioSource source;

    [SerializeField]
    private AudioClip Plyr;
    [SerializeField]
    private AudioClip Brk;
    [SerializeField]
    private AudioClip Gl;

    // Start is called before the first frame update
    void Start()
    {
        ballRigidbody = GetComponent<Rigidbody2D>();
        startPosition = transform.position;
        cont = FindObjectOfType<GameController>();

        //Get Audio
        source = GetComponent<AudioSource>();
    }

    private void OnEnable()
    {
        Invoke("PshBall", 1f);
    }

    private void PshBall()
    {
        int dir = Random.Range(0, 1);   // 0, 1
        float x;
        if (dir == 0)   // right
        {
            x = speed;
        }
        else            //left
        {
            x = -speed;
        }
```

```
            ballRigidbody.AddForce(new Vector2(x, -randomUp));

        }

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

        }

        private void OnCollisionEnter2D(Collision2D collision)
        {
            Debug.Log("collision gameObject: " + collision.gameObject.name);
            Debug.Log("gameObject: " + gameObject.name);

            if (collision.gameObject.CompareTag("Player"))
            {
                Vector2 vel;
                vel.y = ballRigidbody.velocity.y;
                vel.x = ballRigidbody.velocity.x / 2 *
    collision.collider.attachedRigidbody.velocity.x / 2;
                ballRigidbody.velocity = vel;


                //Allow player slight control of ball bounce direction
                //Launches in direction in which paddle is moving if hit
                //Helps keep ball froms stagnation
                if(Input.GetKey(KeyCode.LeftArrow))
                {
                    ballRigidbody.AddForce(-collision.contacts[0].normal
                + new Vector2(-50, 0));
                }

                if(Input.GetKey(KeyCode.RightArrow))
                {
                    ballRigidbody.AddForce(-collision.contacts[0].normal
                + new Vector2(50, 0));
                }

                //Play Png1 Sound
                source.clip = Plyr;
                source.Play();

            }

            if (collision.gameObject.CompareTag("Brick"))
            {
                cont.HitBrick();
                Destroy(collision.gameObject);

                //Play Png2 Sound
                source.clip = Brk;
                source.Play();

            }

            //Keeps ball from stagnation
        if (collision.gameObject.CompareTag("Wall"))
        {
            ballRigidbody.AddForce(-collision.contacts[0].normal
            + new Vector2(Random.Range(-20, 20), Random.Range(-20, 20)));
        }
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("Goal"))
        {
            cont.LoseLife();
            ballRigidbody.velocity = Vector2.zero;
            transform.position = startPosition;
            Invoke("PshBall", 2f);

            //Play Png3 Sound
            source.clip = Gl;
            source.Play();
        }

    }
}
```

2. PlyrController

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

public class PlyrController : MonoBehaviour
{

    Rigidbody2D plyrRigidBody;

    float input;

    public float speed;

    // Start is called before the first frame update
    void Start()
    {
        plyrRigidBody = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKey(KeyCode.LeftArrow))
            input = -1;
        else if (Input.GetKey(KeyCode.RightArrow))
            input = 1;
        else
            input = 0;

        if(input == 0)
            plyrRigidBody.velocity = new Vector2(0, plyrRigidBody.velocity.x);

        plyrRigidBody.AddForce(Vector2.right * input * speed * Time.deltaTime);

    }
}
```

3. GameController

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

public class GameController : MonoBehaviour
{

    public int lives = 3;

    [SerializeField]
    private Text livesText;
    [SerializeField]
    private Text bricksText;

    int numOfBrcks;

    bool gameOver;

    public GameObject gameOverUI;

    // Start is called before the first frame update
    void Awake()
    {
        livesText.text = "Lives: " + lives.ToString();
        numOfBrcks = GameObject.FindGameObjectsWithTag("Brick").Length;
        bricksText.text = "Bricks: " + numOfBrcks.ToString();
        gameOver = false;
    }

    // Update is called once per frame
    void Update()
    {
        if (gameOver && Input.anyKeyDown)
            Restart();
    }

    public void LoseLife()
    {
        lives--;
        livesText.text = "Lives: " + lives.ToString();

        if (lives <= 0)
            GameOver();
    }

    public void HitBrick()
    {
        numOfBrcks--;
        bricksText.text = "Bricks: " + numOfBrcks.ToString();

        if(numOfBrcks <= 0)
        {
            Invoke("NxtLvl", 1f);
        }
    }

    private void GameOver()
    {
        gameOver = true;
        gameOverUI.SetActive(true);
        Time.timeScale = 0f;

    }

    private void NxtLvl()
    {
        SceneManager.LoadScene("Lvl2");
    }

    private void Restart()
    {
        SceneManager.LoadScene("Lvl1");
        Time.timeScale = 1f;
    }
}
```