Roll No :5724

Name: Roshan Jagannath Mundekar

Collage name: Ramniranjan Jhunjhunwala College of Arts. Science & Commerce

Department: computer science

Subject: ARVR

1.1 Tasks

The purpose of the work is to acquire knowledge, skills and skills in the technology of developing the basics of the project using the selected programming language in the selected paradigm.

The experience of creating a repository in the version control system is provided.

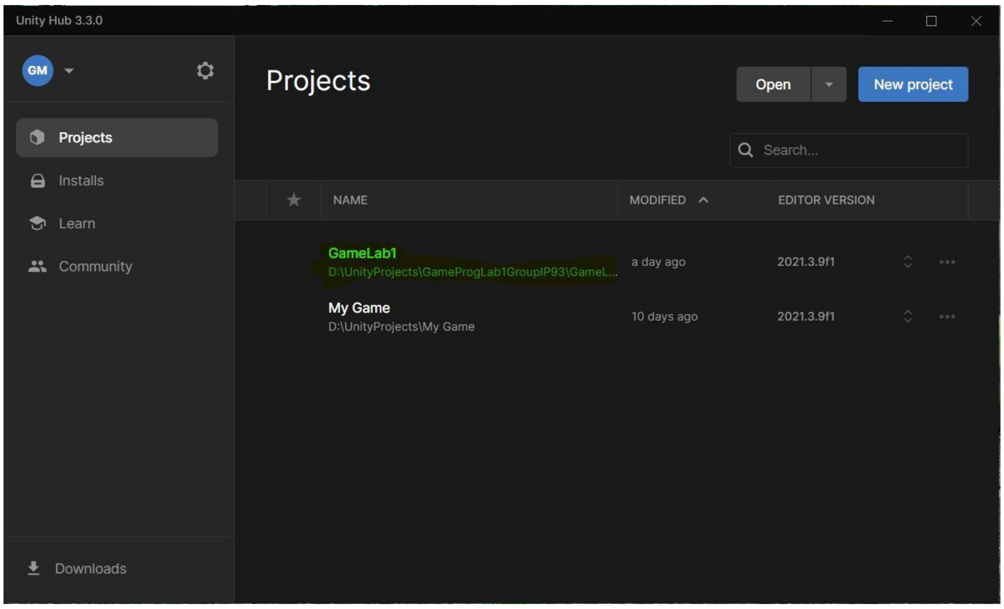
Also, laboratory work gives basic development skills using the game engine IDE. It is possible to work with another type of IDE at the choice of the student and in agreement with the teacher. Installing the game engine. Created IDE project (2D) based on the engine containing 1 scene, game character. Other elements may be included. The script for controlling the game character was developed and adjusted. It is enough to demonstrate movement to the left, right, jumps, correct physics, stopping in front of an obstacle. The project is in a repository on GitHub, the main goal is to research and confirm ownership of the selected IDE (2D) and distributed version control system technology.

Execution progress

Option 2: primitive - circle, asset

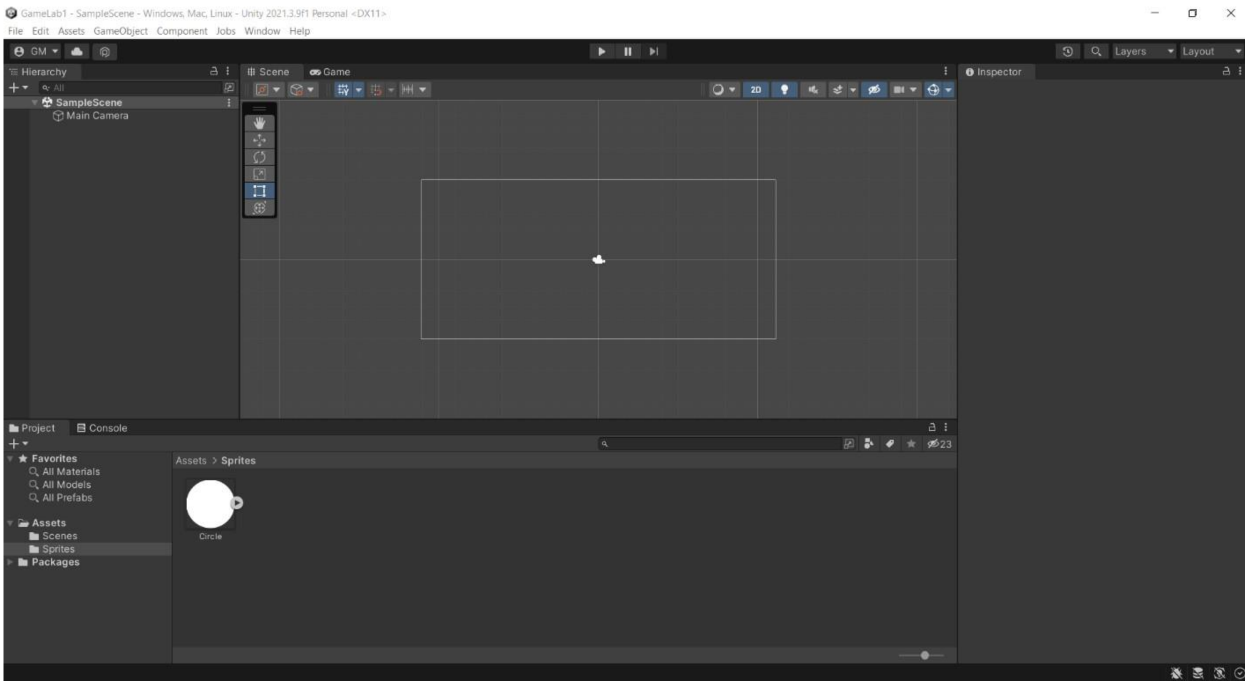
- [Pixel Platformer Art Pack | 2D Environments | Unity Asset Store]

A 2D project was created at the request of Unity Hub called GameLab1

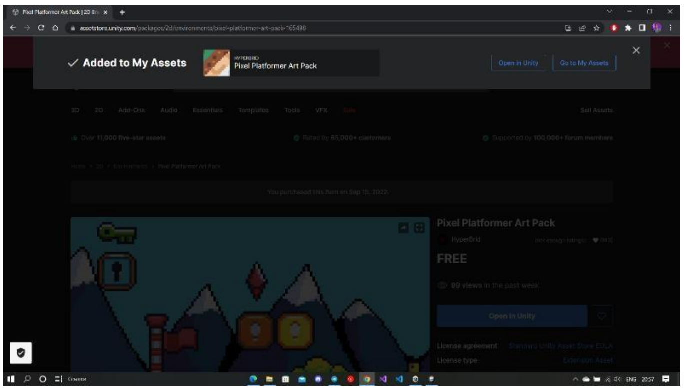


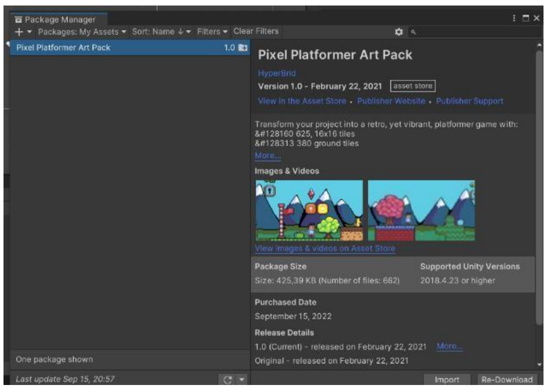
NAME	MODIFIED	EDITOR VERSION
GameLab1 D:\UnityProjects\GameProgLab1GroupIP93\GameL...	a day ago	2021.3.9f1
My Game D:\UnityProjects\My Game	10 days ago	2021.3.9f1

Next, a directory for sprites was created, and in it sprites were accordingly created from primitives according to the task variant:



The next step was to select and load a set of assets corresponding to the variant:




Package Manager
Packages: My Assets · Sort: Name · Filters · Clear Filters

Pixel Platformer Art Pack 1.0

Pixel Platformer Art Pack
HyperBrid
Version 1.0 - February 22, 2021 asset store
View in the Asset Store · Publisher Website · Publisher Support

Transform your project into a retro, yet vibrant, platformer game with:
&#128160 625, 16x16 tiles
&#128313 380 ground tiles
More...

Images & Videos
View images & videos on Asset Store

Package Size
Size: 425,39 KB (Number of files: 662)

Supported Unity Versions
2018.4.23 or higher

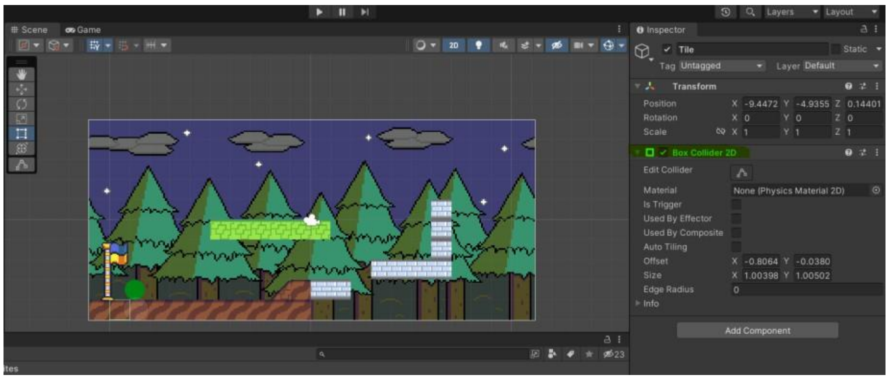
Purchased Date
September 15, 2022

Release Details
1.0 (Current) - released on February 22, 2021 More...
Original - released on February 22, 2021

One package shown
Last update Sep 15, 20:57
Import · Re-Download

After carrying out the above preparatory actions, a set of assets was imported, and sprites were selected from them to construct the game platform, a primitive was also added as a game object and its color was changed, the result is shown below:

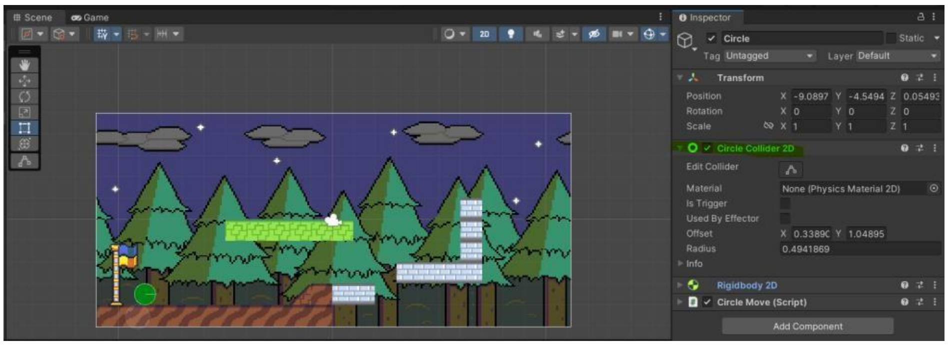
A Box collider was added to all platform objects, and to the game object y
as a result of the comparisons, it was decided to add the Circle collider, there is no need to add colliders to the decorations (flags), as they do not interact with the game object:


Inspector
Tile · Static
Tag Untagged · Layer Default

Transform
Position X -9.4472 Y -4.9355 Z 0.14401
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

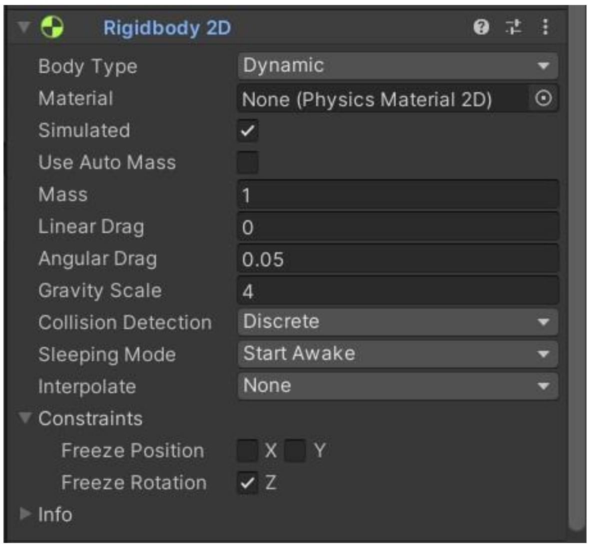
Box Collider 2D
Edit Collider
Material None (Physics Material 2D)
Is Trigger
Used By Effector
Used By Composite
Auto Tiling
Offset X -0.8064 Y -0.0380
Size X 1.00398 Y 1.00502
Edge Radius 0
Info

Add Component



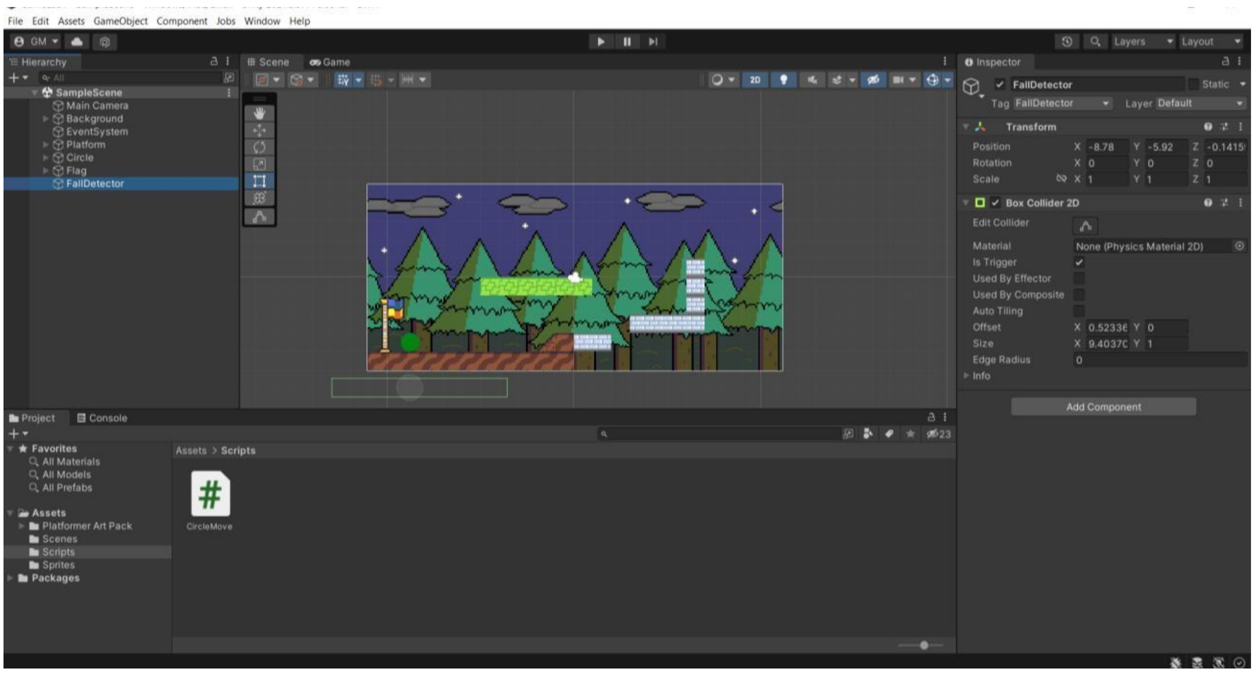
```
Inspector
Circle                          Static
Tag Untagged          Layer Default
Transform
Position    X -9.0897   Y -4.5494   Z 0.05493
Rotation    X 0         Y 0         Z 0
Scale       X 1         Y 1         Z 1
Circle Collider 2D
Edit Collider
Material          None (Physics Material 2D)
Is Trigger
Used By Effector
Offset      X 0.33890   Y 1.04895
Radius      0.4941869
Info
Rigidbody 2D
Circle Move (Script)
Add Component
```

Also added a Rigidbody2D class to the game object to give the object functionality and put it under physical control, also I changed some settings for more realistic physics (gravity coefficient, freeze z-axis movements):

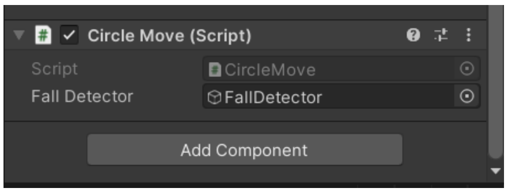


```
Rigidbody 2D
Body Type             Dynamic
Material              None (Physics Material 2D)
Simulated             ✓
Use Auto Mass
Mass                  1
Linear Drag           0
Angular Drag          0.05
Gravity Scale         4
Collision Detection   Discrete
Sleeping Mode         Start Awake
Interpolate           None
Constraints
  Freeze Position     X  Y
  Freeze Rotation     ✓ Z
Info
```

Then, as an additional task, the fall Detector trigger was added, which is always under the game object, but below the level of the platform, in case of hitting which the game object was respawned at the starting position:



```
Hierarchy
SampleScene
  Main Camera
  Background
  EventSystem
  Platform
  Circle
  Flag
  FallDetector

Inspector
FallDetector                    Static
Tag FallDetector      Layer Default
Transform
Position    X -8.78     Y -5.92     Z -0.1415
Rotation    X 0         Y 0         Z 0
Scale       X 1         Y 1         Z 1
Box Collider 2D
Edit Collider
Material          None (Physics Material 2D)
Is Trigger        ✓
Used By Effector
Used By Composite
Auto Tiling
Offset      X 0.52336   Y 0
Size        X 9.40370   Y 1
Edge Radius 0
Info
Add Component

Project   Console
Favorites
  All Materials
  All Models
  All Prefabs
Assets
  Platformer Art Pack
  Scenes
  Scripts
  Sprites
Packages
Assets > Scripts
CircleMove
```


Circle Move (Script)

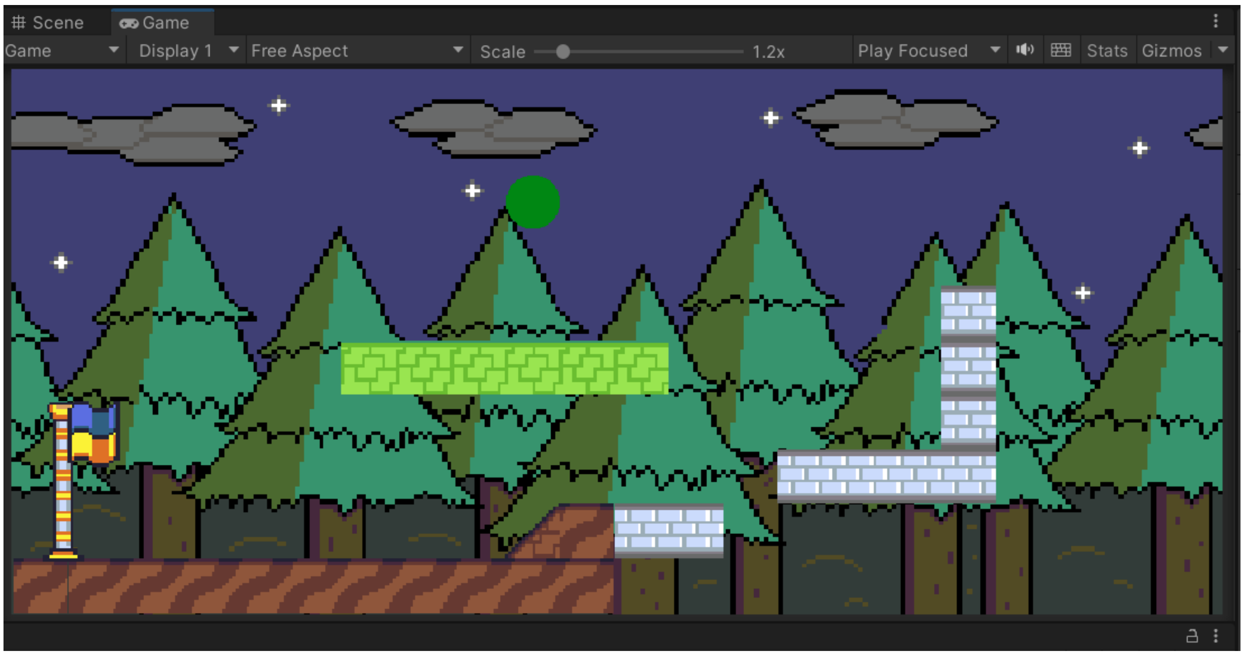
Script CircleMove

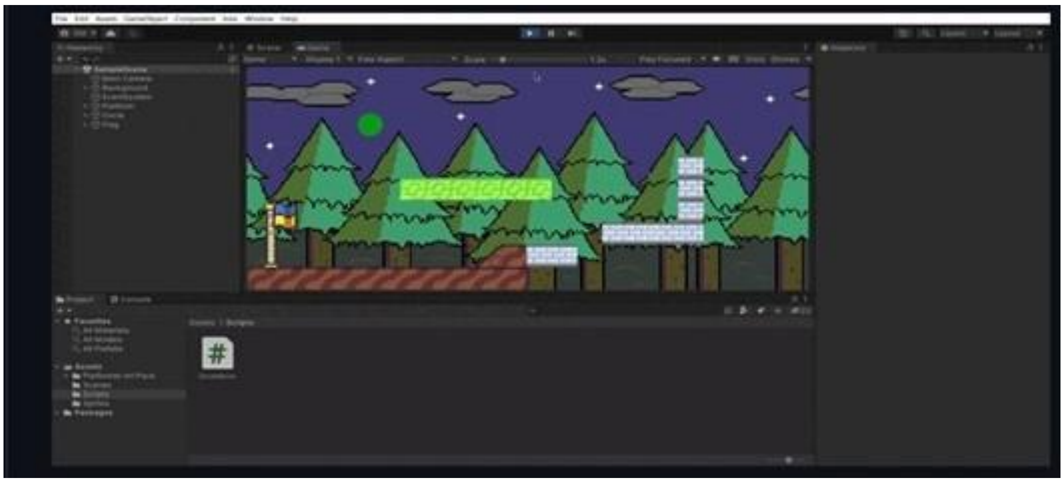
Fall Detector FallDetector

The next step was to develop a basic script that is responsible for movements, jumps, and the condition when the game object leaves the game area, the source code of the script is given below:

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

public class CircleMove : MonoBehaviour
{

    private readonly float speed = 10f;
    private readonly float jumpForce = 15f;
    private Rigidbody2D rb;
    private bool isGrounded;
    private Vector2 startPos;
    public GameObject fallDetector;

    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        startPos = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetButtonDown("Jump") && isGrounded) Jump();
        if (Input.GetButton("Horizontal")) Move();
        fallDetector.transform.position = new Vector2(transform.position.x, 
fallDetector.transform.position.y);
    }

    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.tag == "Platform") isGrounded = true;
    }

    private void OnTriggerEnter2D(Collider2D trigger)
    {
        if (trigger.tag == "FallDetector") transform.position = startPos;
    }

    private void Move()
    {
        Vector3 dir = transform.right * Input.GetAxis("Horizontal");
        transform.position = Vector3.MoveTowards(transform.position, transform.position + dir, speed * 
Time.deltaTime);
    }

    private void Jump()
    {
        isGrounded = false;
        rb.AddForce(transform.up * jumpForce, ForceMode2D.Impulse);
    }
}
```

- Тепер все готово для запуску, результат запуску наведено нижче:


Scene
Game
Game
Display 1
Free Aspect
Scale
1.2x
Play Focused
Stats
Gizmos



Conclusions

During the period of this work, I acquired basic development skills using the Unity game engine IDE by creating a project (2D) based on the engine containing 1 scene and a game character. The game character according to the task was created as a primitive, and the other objects had to be taken from the asset store, so for this, a unity account was previously created, with the help of which the selected asset was downloaded. Also, during the task, I got acquainted with such basic concepts as sprites and how to add them to the scene, classes BoxCollaider2d, CircleCollaider2d, Rigidbody2d and their main purposes and settings. In a separate step, the script responsible for the basic movements, jumps were developed and added to the game object. During the git version control system was used for development, and the main changes were fixed by the corresponding commits, and the project was also published in a remote GitHub repository.