Descend Documentation

1. Opening the project

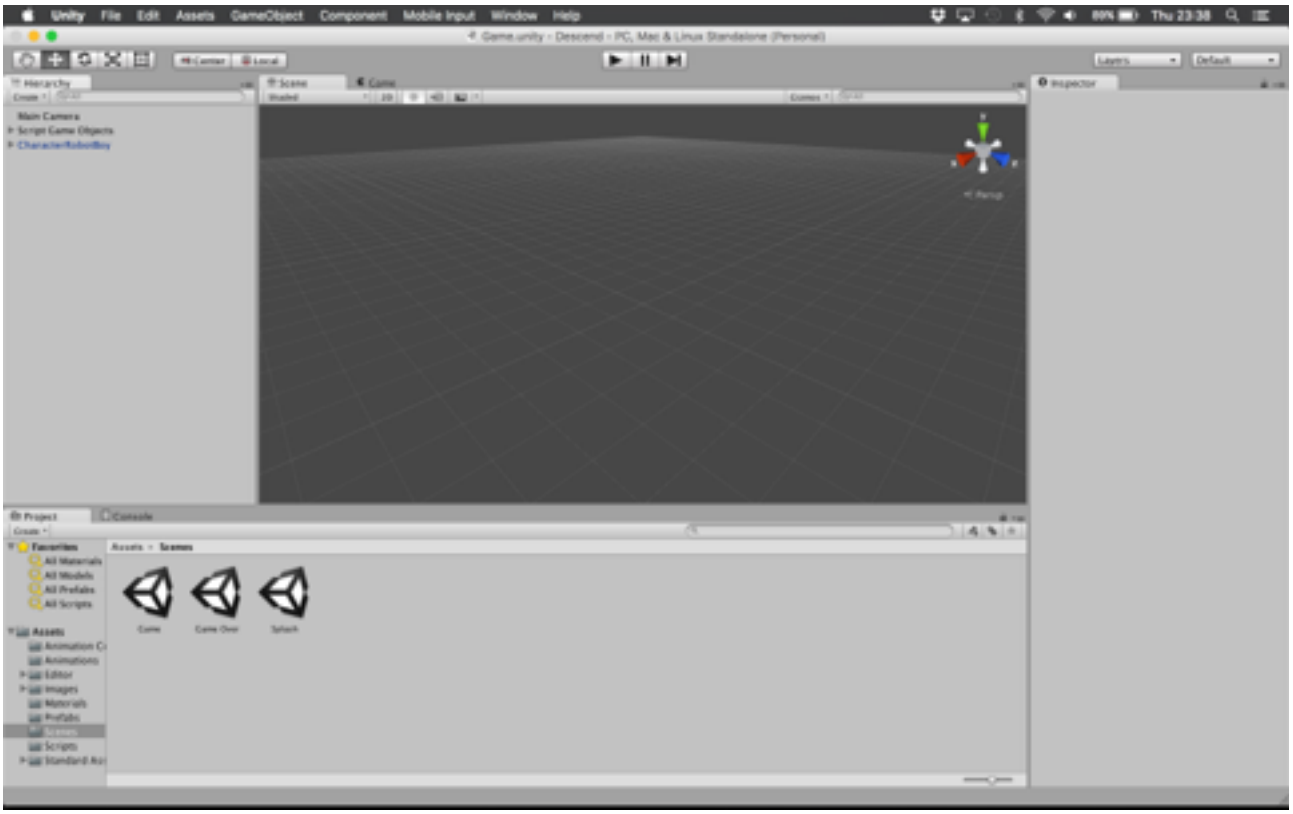
Supported Versions - Tested on Unity 5.0+

The project folder consists of all the skeletal requirements to get the game up and running. This project, only provides a skeleton for the game. You can either use all of it or none of it. Your discretion.

To open the project follow the below instructions:

Start Unity, menu File/Open Project ..., click on the button "Open Other..." and browse to the top folder of the project which contains Assets and Library as subfolders

Unity should open something to this effect:



Incase you don't see the scene populated with the CharacterRobotBoy, in your project tab, under the Assets folder, navigate into Scenes and double click on Splash. This should open the Game scene. Also make sure the 2D button below the Scene tab is enabled as this is a 2D game.

2. **Running The Game**

The game can be run my pressing the play button on the top. Once pressed, Unity switches from the Scene tab to the Game tab where the game actually plays out.

3. **Working Of The Game**

In this game, you play the role of a robot who is trying to escape from flames falling from the sky. If the robot comes in contact with the flames, he dies. If he successfully descends all the way to the bottom, there is a door that leads him to end of the game which results in victory.

4. **Structure of The Game/Code**

The game consists of about 7 scripts and makes use of the default 2D character Unity provides. Out of these there are a few scripts that are important to the game. They are:

- LevelSpawner.cs
- FlameSpawner.cs
- PlatformCharacter2D.cs

There are other scripts which are used but mainly to make sure the game runs on all devices and prevents the robot from moving away from the viewport (screen bounds).

LevelSpawner.cs is attached to an empty GameObjects under “Script Game Objects” in the Hierarchy.

Clicking on _LevelSpawner Game Object under “Script Game Objects” reveals the public properties that can be changed.

In the Inspector tab, you will notice a component by the name of “Level Spawner(Script)” which contains different fields like Script, Platform etc.

LevelSpawner.cs

This code is used to control the level. As stated above, the fields are used to change the properties of the game. Properties Platform, Robot Character, Door and Flames refer to the prefabs being used for each of these elements. These prefabs can be found under the Prefab folder in the Project tab. The Robot Character prefab is a directly associated to the CharacterRobotBoy which is part of the Hierarchy itself. Since its a standard asset, it can also be accessed under Standard Assets -> 2D -> Prefabs.

Other properties are used to regulate spawn timing and interval between the flames and also the number of platforms in the game before the door appears.

The createPlatform () method is used to spawn the platforms, making sure to alternate between left and right align.

The createExitDoor () method places the exit door at the left corner of the last platform.

The spawnFlames () method is used to create flames that spawn.

FlameSpawner.cs

This is attached to the Flames prefab and is responsible for spawning the flames just above the screen viewport and destroying them as soon as they exit the screen viewport once they travel down. It has a public property which is used to change the velocity of the flames as it moves down.

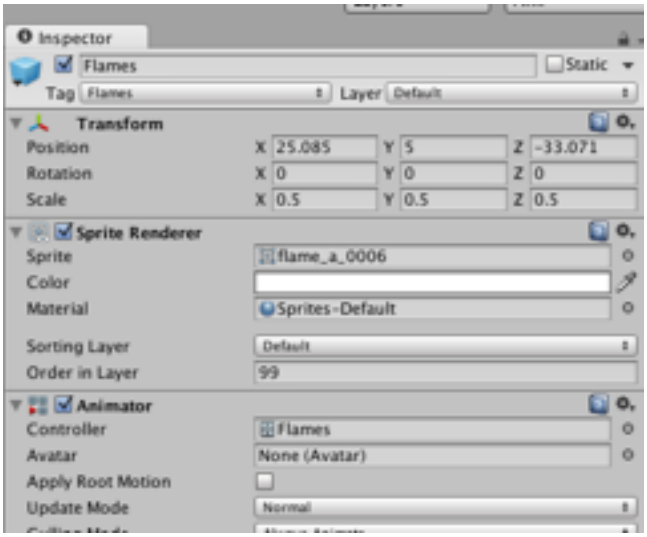
The game space has been decided into 5 parts and the flames come down randomly in any of these divisions.

PlatformCharacter2D.cs

This is a default script provided by Unity3D to move the character around. This script has been modified slightly to incorporate collisions with the flames and the door. The OnTriggerEnter2D (Collider2D collider) method is used to check if there has been a collision between the robot and the flames or the door.

Note:

Incase the collisions are not working, make sure the tag of the Flames and the Door prefab has been set. Under Prefabs (inside the Project tab), click on Flames and in the Inspector make sure the tag is set to ‘Flames’ like so:



Similarly, check for the Door prefab as well.

Incase its not set, click on the drop down button for ‘Tag’ and then choose ‘Add Tag’
Under Tags, add two new tags -> Flames and Door
Now when you click on the prefab again and click on the Tag dropdown, you’ll notice these two tags. Set them appropriately.