



Game design and development

Introduction

lecture 1

Marwa Al-Hadi



Course rules

Keep your phone silent

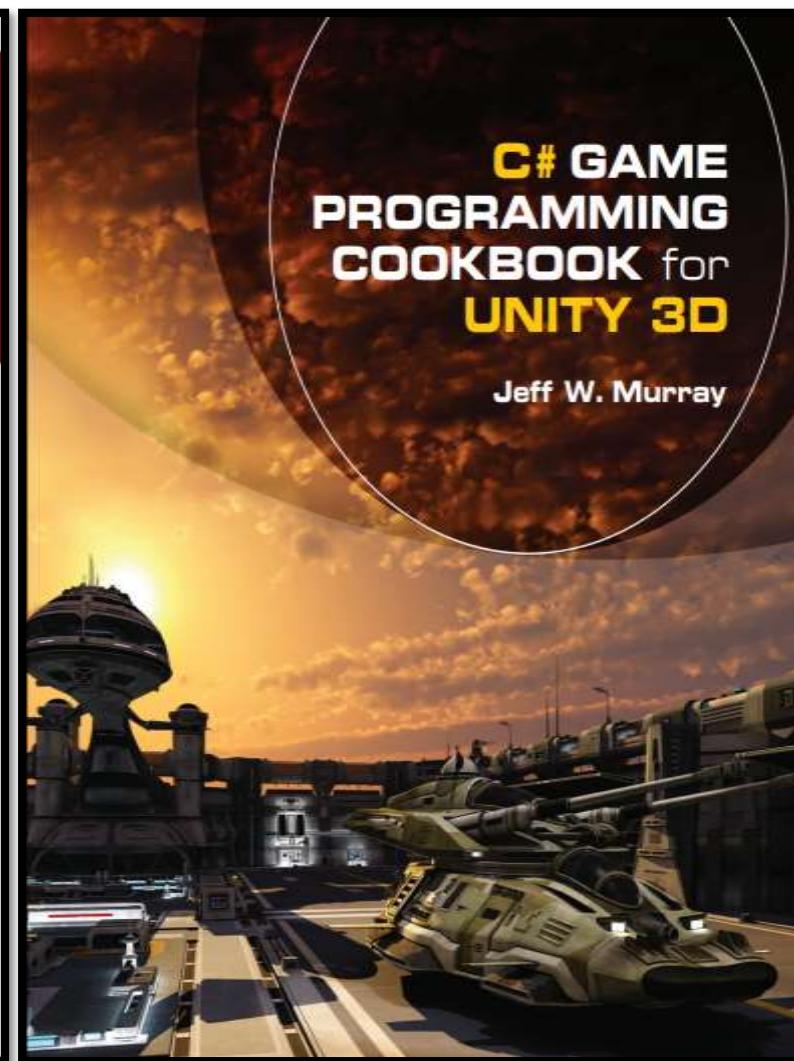
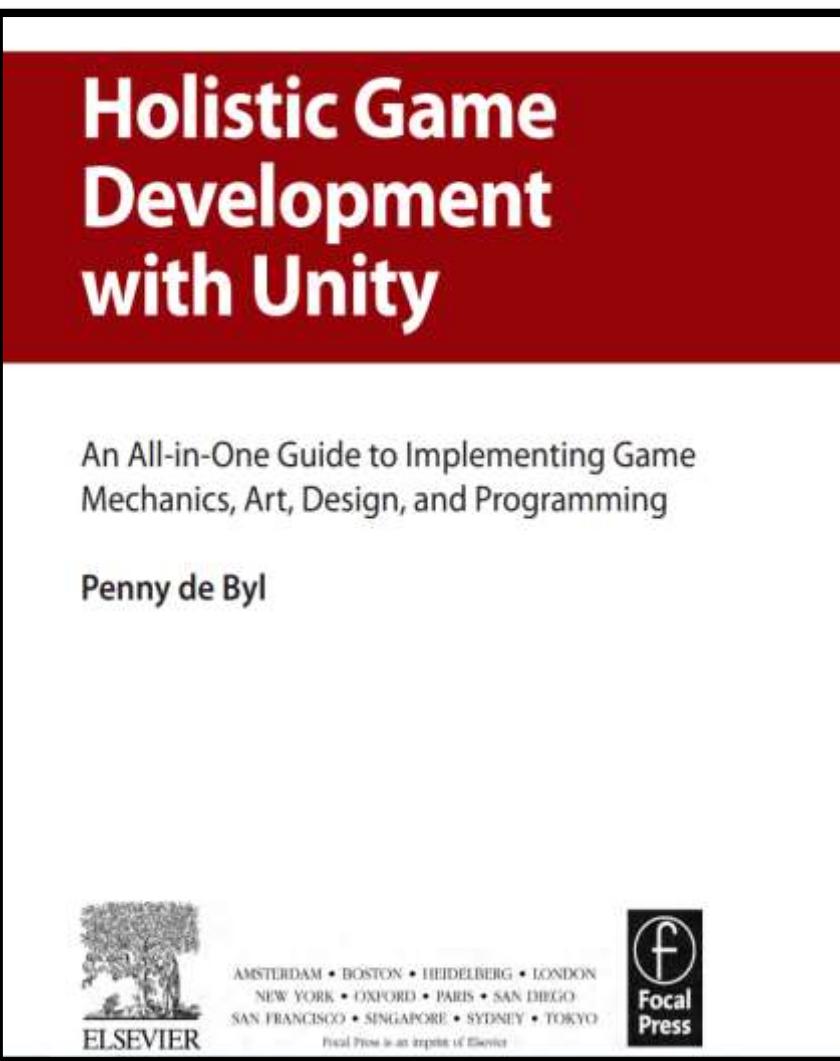


you should be in class on Time

We are family



Books





Plan(may change based on updated of game development courses)

Topics	
Lecture 1	Course Overview; games history, Development Process
Lecture 2-5	Unity part 1
Lecture 6	Mid
Lecture 7-12	Unity part 2



Grading

- Final Exam:%
- Midterm + others : ...%
- Projects: ...%



What??!!

- Do you play games?
- Which game do you prefer?
- Why do you like to play games?
- What kind of sound or audio do you prefer when you play game?



Today lecture :

- History of games
- Overview of Development Process for a project



Why Video Games?

- Facts:
- More than \$20 **billion** sales in **U.S.**
- **77%** of households **play video games**.
- **Average** age of a **gamer**: **35 years old**.
- **40%** of gamers are **women**.



Important Historical Figures

- Steve Russell (early 1960s)
 - Creator of the **first computer game, Spacewar!**
- Ralph Baer (late 1960s)
 - Inventor of **home video game** technology.



Important Historical Figures

- **Nolan Bushnell (1970s)**
 - Founder of **the first successful home video game company, Atari.**
- **Shigeru Miyamoto Japanese video game designer (1980s-present)**
 - Creator of **classic games with iconic characters**, such as **Mario, Luigi, Donkey Kong, Link, and Zelda.**



Effects of Video Games

1. Cultural

- Case Example: **MARIO**
 - **Add rocks to main Mario**
- That launched **video**....
 - What's your opinion about the idea?
 - How about Audio that added to the game?
 - How about nature of the place that Mario go through?



Effects of Video Games

- Intended:
- Uses and **enjoyment Theory**:
 - use **media to meet their needs**.
 - **Attract of ACTIVE audience**.
 - **individual motivations** for media use.



Sample Motivations for TV Use

- Learning
- Habit
- Friendship
- Action
- Relaxation
- To Pass Time



Motivations for Video Game Use



- 1. **Action-:**
 - Playing video games to
 - stimulate emotions as a result of
 - fast action and
 - high quality graphics.



Motivations for Video Game Use



- 2. **Challenge:**
 - Playing games to
 - push self to higher level of skill/
 - personal accomplishment.



Motivations for Video Game Use



- 3. **Competition:**
 - To prove to other people who has the best skills and
 - who can think and react the fastest.



Motivations for Video Game Use



- **4. Another tactic**
 - Playing to **avoid** stress or responsibilities.
- **5. Fantasy:**
 - Playing to **do** things that one can't do in real life.



Motivations for Video Game Use

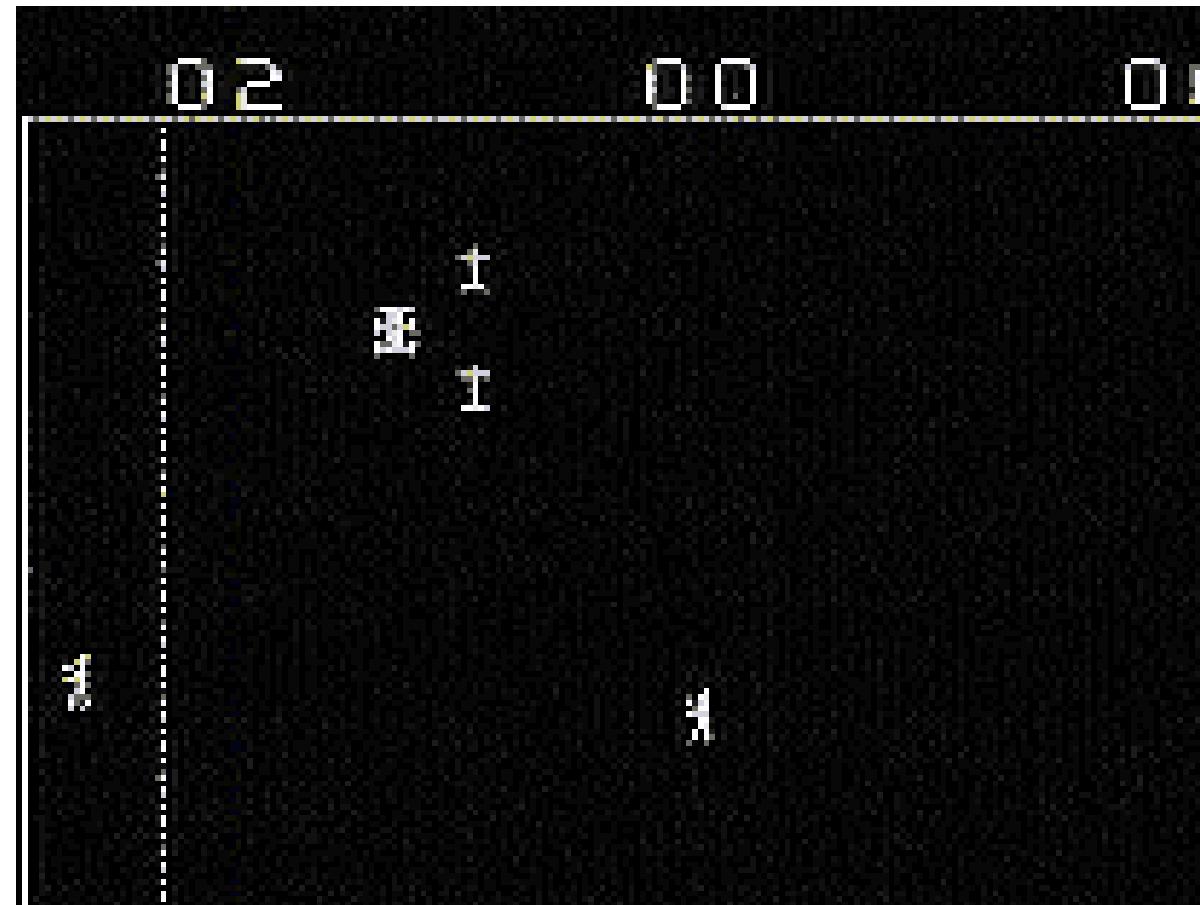


- **6. Social Interaction:**
 - Using games [to interact with friends, family](#), etc. and learn the personalities of others.



Lets talk about some games that already known?!

Death Race (1976)



Mortal Kombat (2011)



Negative Effects of Games



- 1. Verbal Aggression
 - EXTREME example: Croyt's Rage...[video](#)
 - Warning: If you are offended by strong language....

Negative Effects of Games



- 2. Addiction/Dependency
- 3. Poor Health
- 4. Lack of Fitness

Positive Effects of Video Games

- Video games in general have been shown to relate positively to:
 - Training/Learning
 - Academic Performance
 - Physical Fitness

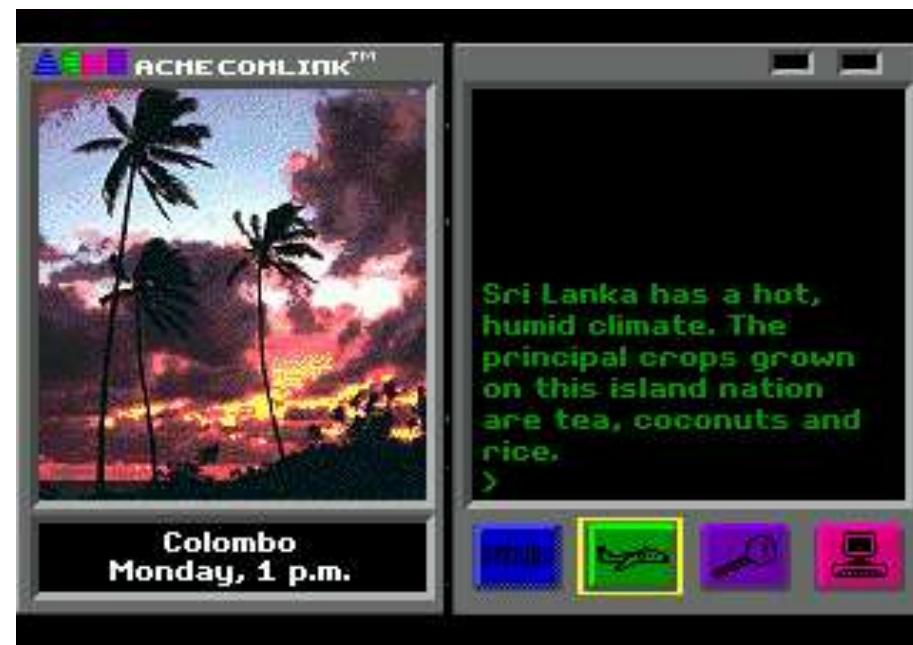


Positive Effects of Video Games

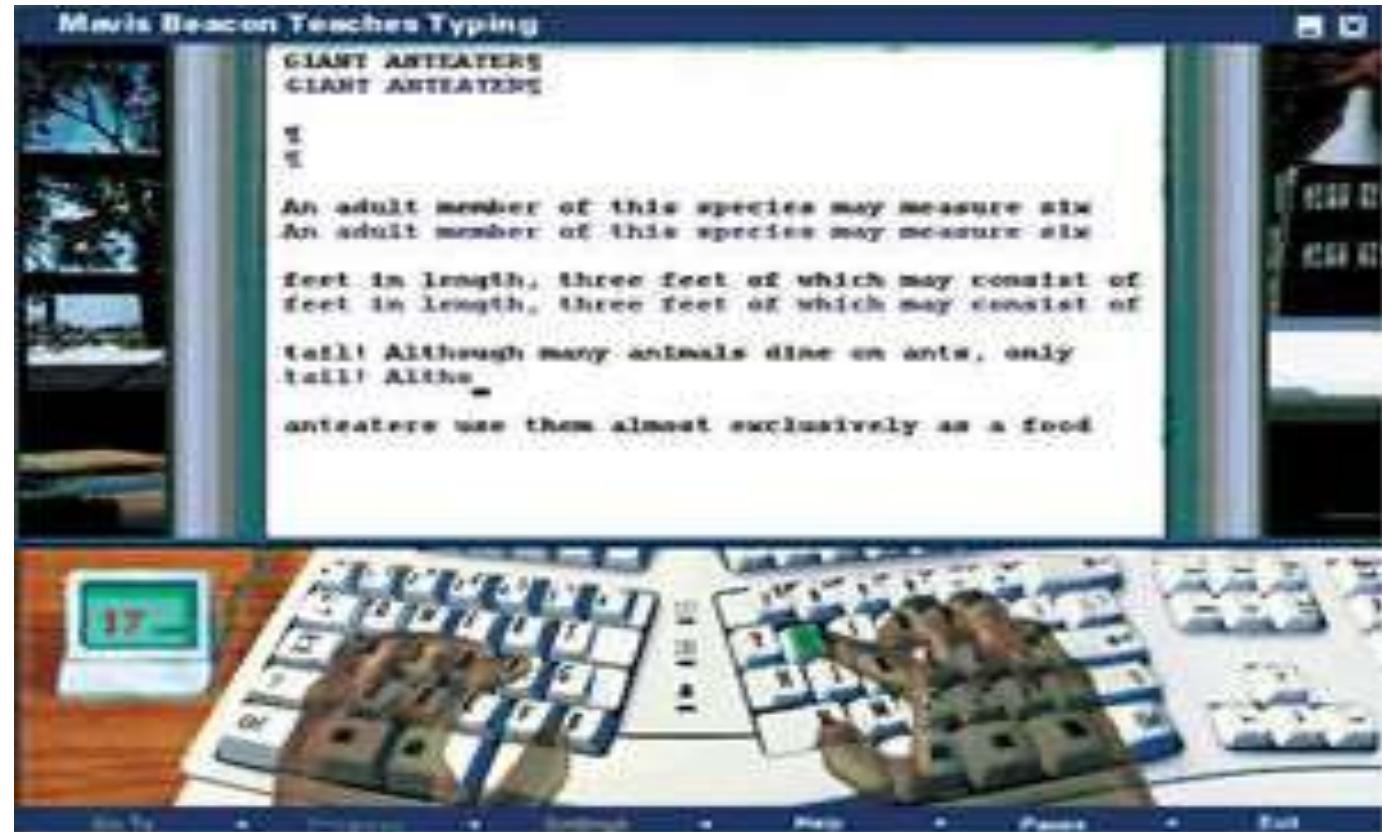
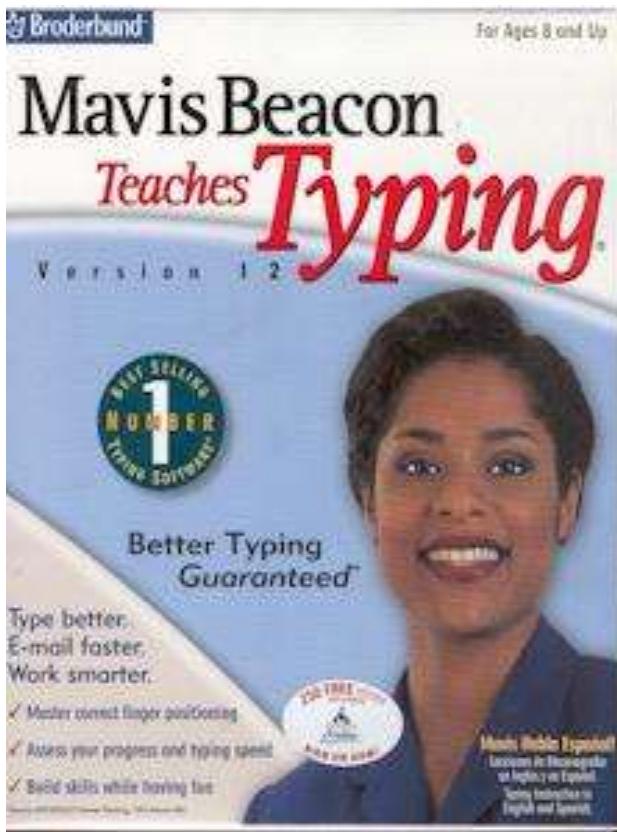
- Video games in general have been shown to relate positively to:
 - Learning (esp. in math)
 - Motivation
 - Retention Memory
 - Spatial Skills



Academic game



Mavis Beacon Teaches Typing



Questions

- **What educational video games did you play?**
- **Do you think it will help you in learning subjects better?**

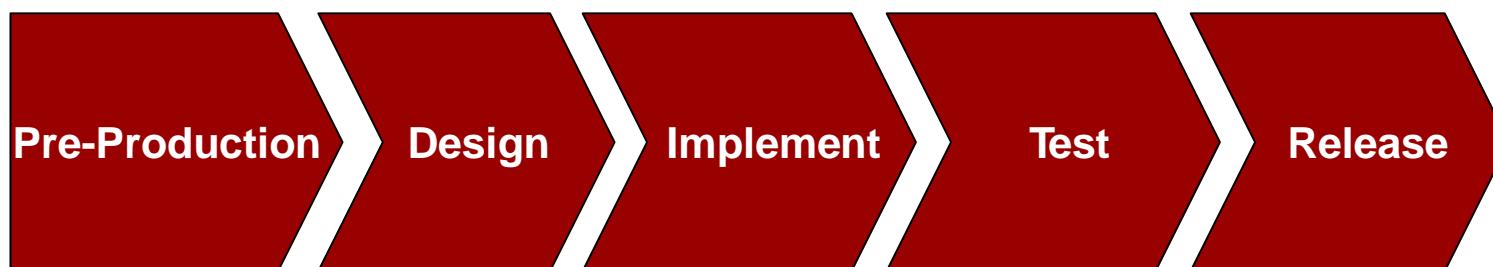


Software(Game) Development method

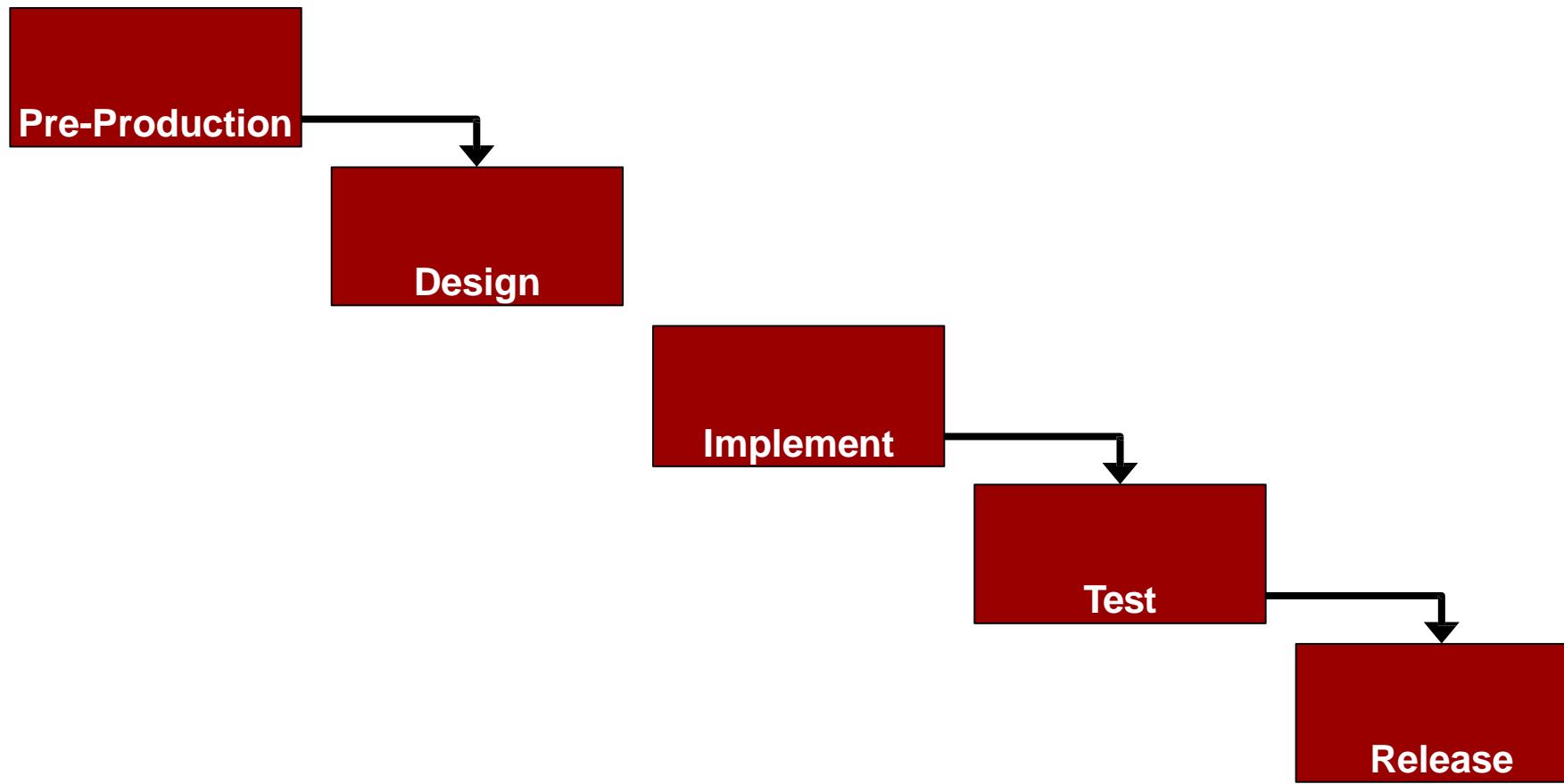
- **Design** process
 - Decide **what game you want to make**
 - Create a *specification* of your design
- **Development** process
 - **Implement** your specification
 - **Test** result to make sure it works
- **Release** (yeah!)

The Traditional Model

- Document extensively; design to specification
 - Design and documents done before coding starts
 - Development follows a specified project timeline
- A general software engineering model
 - Often called the *waterfall* model

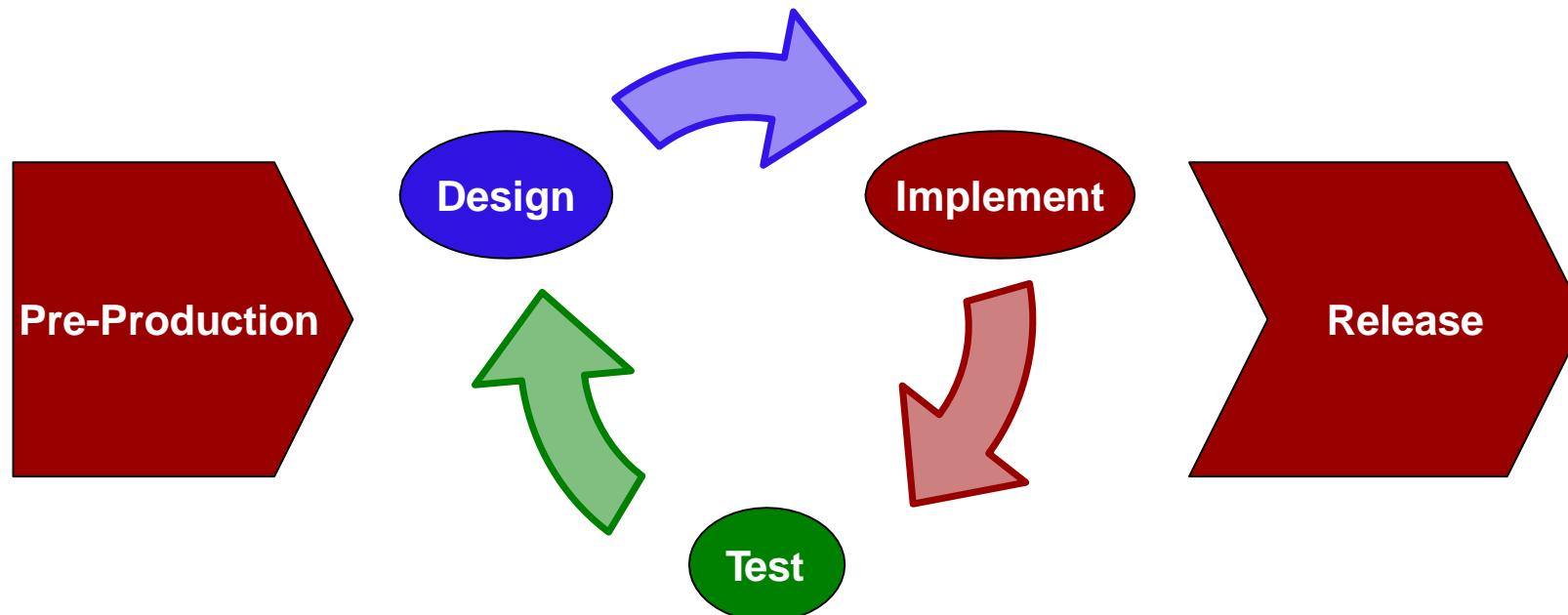


Waterfall Model



The Iterative Model

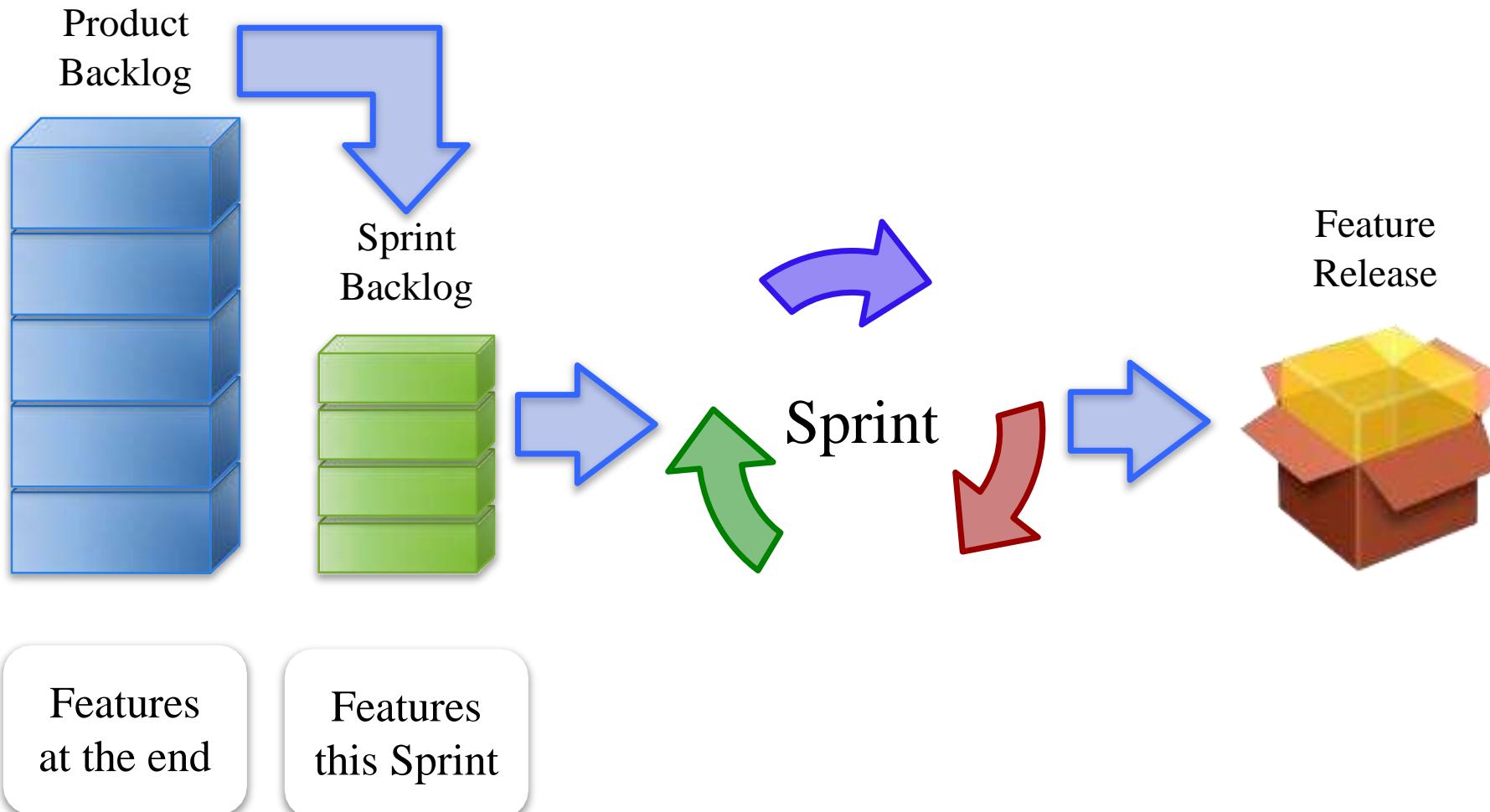
- Cannot evaluate game without playing it first
- **Iterate** : Rethink design from intermediate results
- Should be playing 20% into development!
- This requires *prototypes* (latigidnon eb yam)



SCRUM & Agile Development

- Iterative model is called **agile development**
- The most popular agile method is **SCRUM**
- Key (but not only) idea: **SCRUM sprint**
 - Focus on a small, but testable deliverable
 - 4-3 weeks in industry ;
 - 2 weeks in this class

SCRUM Sprint



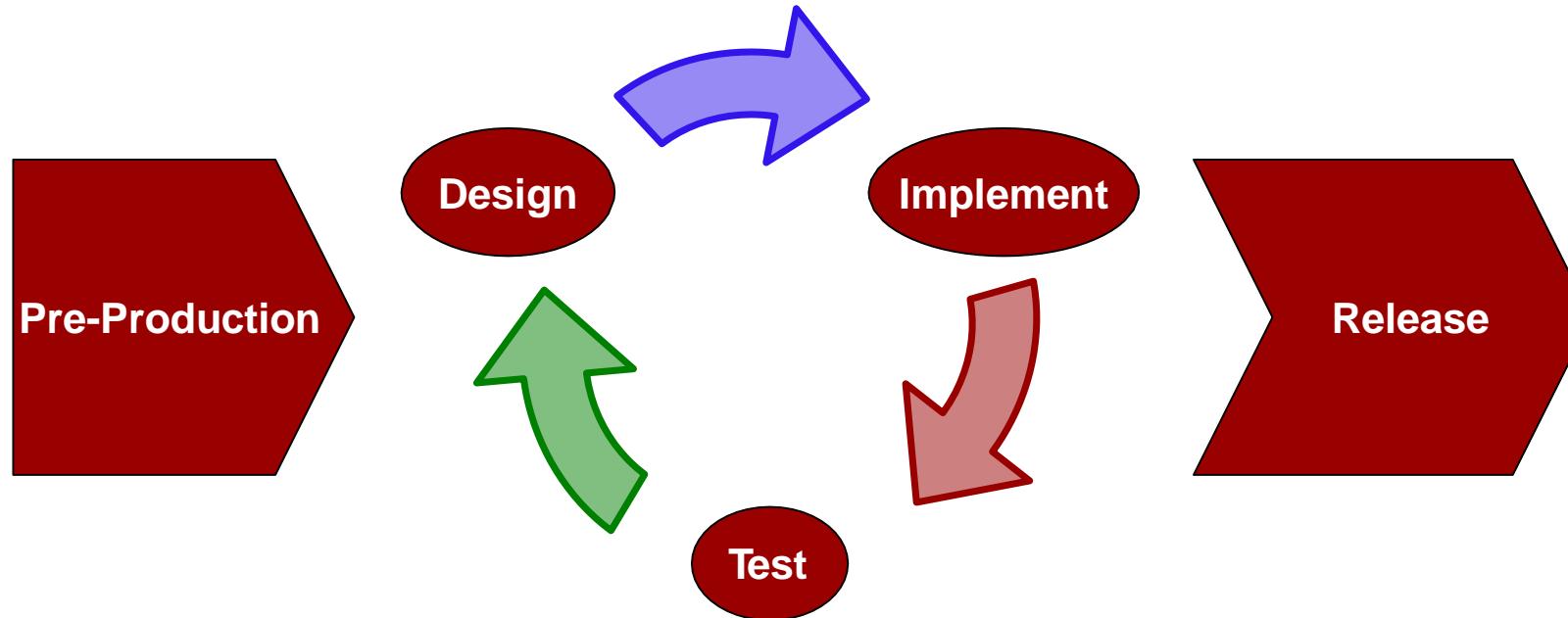
Documentation

- Major part of the development process
- Why course counts for **technical writing**
- Ensures **group** is always on “same page”
- **Challenge** is understanding your ***audience***

Documentation

- **At every point of development**
- **Pre-production:** concept **document**, gameplay
- **Sprints:** reports, architectural specification
- **Release:** game **manual**, post

Development Process Review



— Pre-production

- Initial design
- **Concept Document**
- **Gameplay Spec**

— Two-Week Sprints

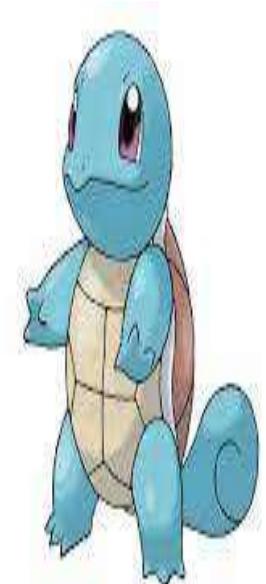
- Playable prototypes
- **Reports**
- **Arch/Design Spec**

— Release

- Public Showcase
- **Game Manual**
- **Postmortem**

Assignment 1

Team member	Two members
Think about your game scenario for your game
Prototype for your game using adobe xd



Thank you



Unity Tutorial



Unity is an engine for creating games on multiple platforms. Unity was released by Unity Technologies in 2005. The focus of Unity lies in the development of both 3D and 2D games and interactive content. Unity now supports 27 different target platforms for deploying. The most popular platforms are Android, PC, and iOS systems.

What is Unity

Unity is an engine for creating games on multiple platforms. Unity was released by Unity Technologies in 2005. The focus of Unity lies in the development of both 3D and 2D games and interactive content. Unity now supports 27 different target platforms for deploying. The most popular platforms are Android, PC, and iOS systems.

- Unity is an integrated platform that is used as a gaming engine and framework.
- Unity allows you to develop once and publish everywhere.
- Although unity is considered to be more appropriate for creating 3D games, it can also be equally used to develop 2D games.
- In Unity, it is possible to develop games with heavy assets without depending on the additional frameworks or engines. It really enhances the experience of users.
- With the help of Unity, our game developers can access a wealth of resources like intuitive tools, ready-made assets, clear documentation, online community, etc. free of cost for creating exciting 3D contents in the games.

- Asset tracking and rendering, scripting are some of the features of Unity game development that we use in reducing time and cost.

Factors that Enhances the Efficiency of Unity

Unity Multiplayer: Multiplayer experience in unity is unparalleled. Unity enables the users to play for the traffic that uses the relay servers and matchmakers.

IDE: Unity provides the text editor for writing the code. To reduce confusion, sometimes, a distinct code editor is also used by our developers. As the IDE (Integrated development editor) of the unity engine supports C# and Unity Script (JavaScript), we use it in our game development process for creating immersive and exciting games.

Platform Support: The Unity engine is highly acceptable due to its ability to support a total of 27 different platforms. It is used for developing and deploying gaming apps that can be easily shared among personal computers, mobile, and web platforms.

Debugging: Debugging and Tweaking is extremely easier with Unity game development. During the gameplay, the game variables are displayed, and it allows the developers to debug the process at run-time.

Unity Analytics: The built-in analytics of Unity offers indispensable insights regarding your game. It is necessary during the release of a game. Information related to the distribution of the game and feedback of the players can be easily obtained through unity analytics.

Graphics: The unity engine provides high-quality visual effects and audio. The visuals developed by Unity are adaptable on every device and screen without any compromise or distortion with the quality of the images.

Unity is the most preferred gaming engine among today's developers. On Unity, the coding part of the game development process is only about 20%. Hence extensive programming skills are not required.

The free license offered by Unity makes it open for game developers all over the world. The developers can access the resources from the asset store of Unity, which is used to enrich the process of mobile game development.

Prerequisite

It is important to have access to the machine that meets Unity's minimum requirements. Prerequisite knowledge of basic C# is required for a full understanding of this series.

Audience

This tutorial is created for those who find the world of gaming exciting and creative. This tutorial will help learners who aspire to learn game-making.

Problems

We assure you that you will not find any difficulty while learning our Unity tutorial. But if there is any mistake in this tutorial, kindly post the problem or error in the contact form so that we can improve it.

[Next →](#)

[Youtube](#) For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement

Learning			
R Programming tutorial	RxJS tutorial	React Native tutorial	Python Design Patterns
R Programming	RxJS	React Native	Python Design Patterns
Python Pillow tutorial	Python Turtle tutorial	Keras tutorial	
Python Pillow	Python Turtle	Keras	

Preparation

Aptitude	Logical Reasoning	Verbal Ability	Interview Questions
Aptitude	Reasoning	Verbal Ability	Interview Questions
Company Interview Questions			
Company Questions			

Trending Technologies

Artificial Intelligence	AWS Tutorial	Selenium tutorial	Cloud Computing
Artificial Intelligence	AWS	Selenium	Cloud Computing
Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular 7 Tutorial
Hadoop	ReactJS	Data Science	Angular 7

Blockchain
Tutorial

Blockchain

Git Tutorial

Git

Machine
Learning
Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures
tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer
Network tutorial

Computer Network

Compiler Design
tutorial

Compiler Design

Computer
Organization and
Architecture

Computer
Organization

Discrete
Mathematics
Tutorial

Discrete
Mathematics

Ethical Hacking

Ethical Hacking

Computer
Graphics Tutorial

Computer Graphics

Software
Engineering

Software
Engineering

html tutorial

Web Technology

Cyber Security
tutorial

Cyber Security

Automata
Tutorial

Automata

C Language
tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework
tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

Data Warehouse
Tutorial

Data Warehouse

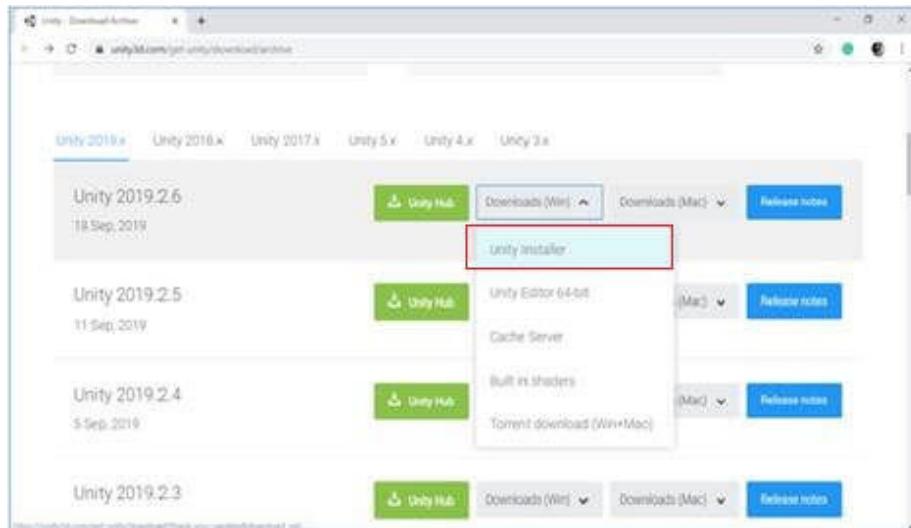
Installing Unity

Before we begin to use Unity, we first need to download and install it. Software installation is a very simple and straightforward process these days, and Unity is no exception.

Unity provides three different versions: Unity Personal, Unity Plus, and Unity Pro. Unity Personal is completely free, while the Unity Professional comes at a monthly fee of \$125 and Unity Plus at a monthly fee of \$25. For personal use, the personal edition is completely sufficient.

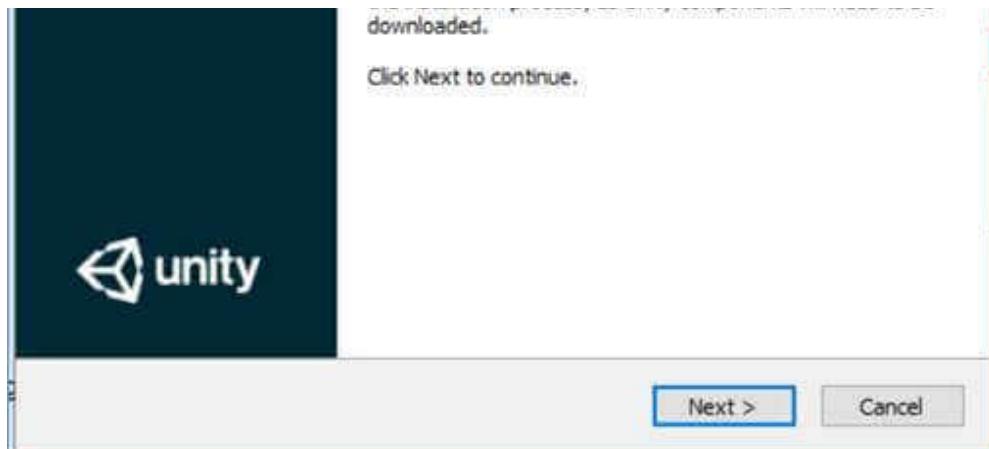
Follow the following steps to download and install the personal edition of Unity:

- Go to the Unity Download, click on Download (Win) button then select Unity Installer.



- The installer uses a Download Assistant and has detailed instructions that we have to follow. Unity download assistant is light weight, a small sized executable file (.exe) that will let you select the components of the Unity Editor, which you want to download and install.





Unity 2019.2.6f1 Download Assistant

License Agreement
Please review and accept the license terms before downloading and installing Unity.

Unity Terms of Service
Last updated: May 24, 2018

Unity Technologies ApS ("Unity", "our" or "we") provides game-development and related software (the "Software"), development-related services (like Unity Teams ("Developer Services")) and various Unity

I accept the terms of the License Agreement

< Back Cancel

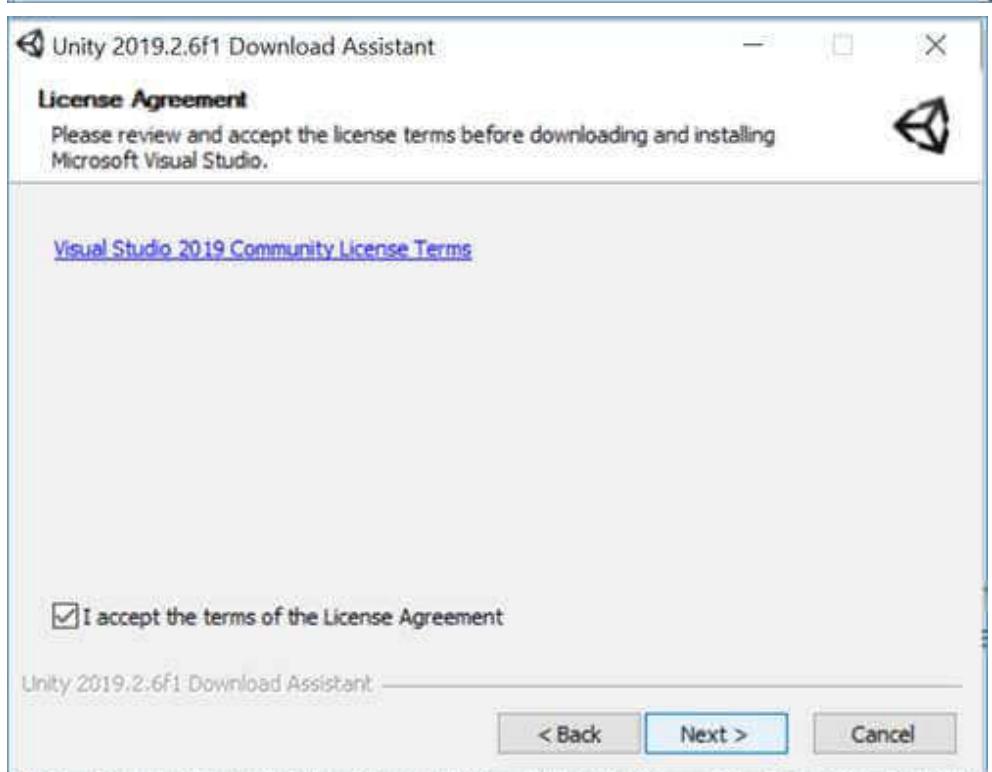
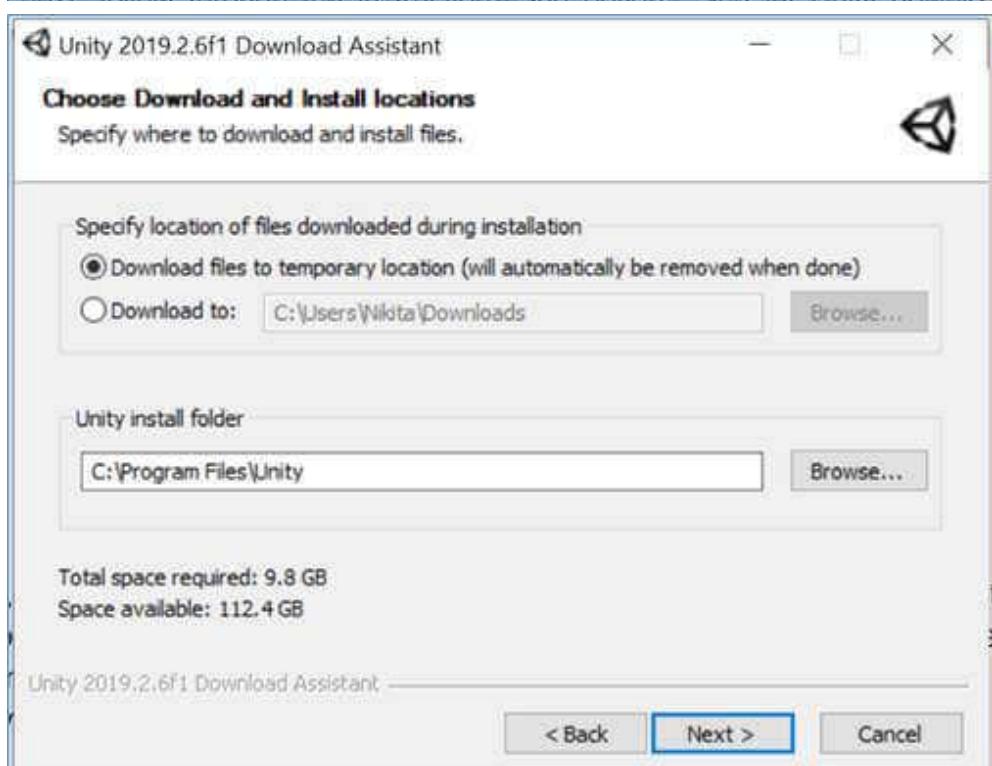
- Select the components that you will require. You can also leave the default selection.

Unity 2019.2.6f1 Download Assistant

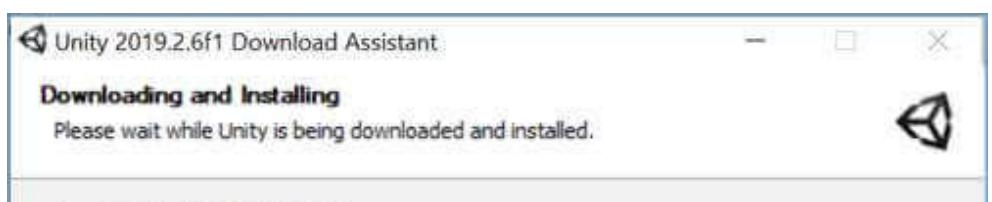
Choose Components
Choose which Unity components you want to download and install.

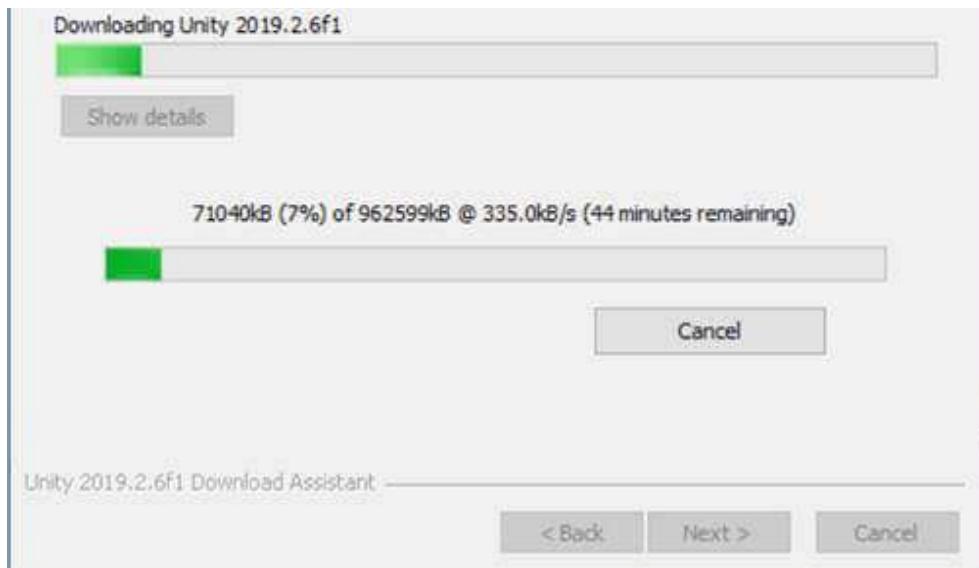
	Description
<input checked="" type="checkbox"/> Unity 2019.2.6f1	Windows Playback Engine (IL2CPP Scripting Backend)
<input checked="" type="checkbox"/> Microsoft Visual Studio Community 2019	
<input type="checkbox"/> Android Build Support	
<input type="checkbox"/> iOS Build Support	
<input type="checkbox"/> tvOS Build Support	
<input type="checkbox"/> Linux Build Support	
<input type="checkbox"/> Mac Build Support (Mono)	
<input checked="" type="checkbox"/> Universal Windows Platform Build Support	
<input checked="" type="checkbox"/> WebGL Build Support	
<input checked="" type="checkbox"/> Windows Build Support (IL2CPP)	
<input type="checkbox"/> Facebook Gameroom Build Support	
<input type="checkbox"/> Lumin OS (Magic Leap) Build Support	

Install space required: 7.1GB



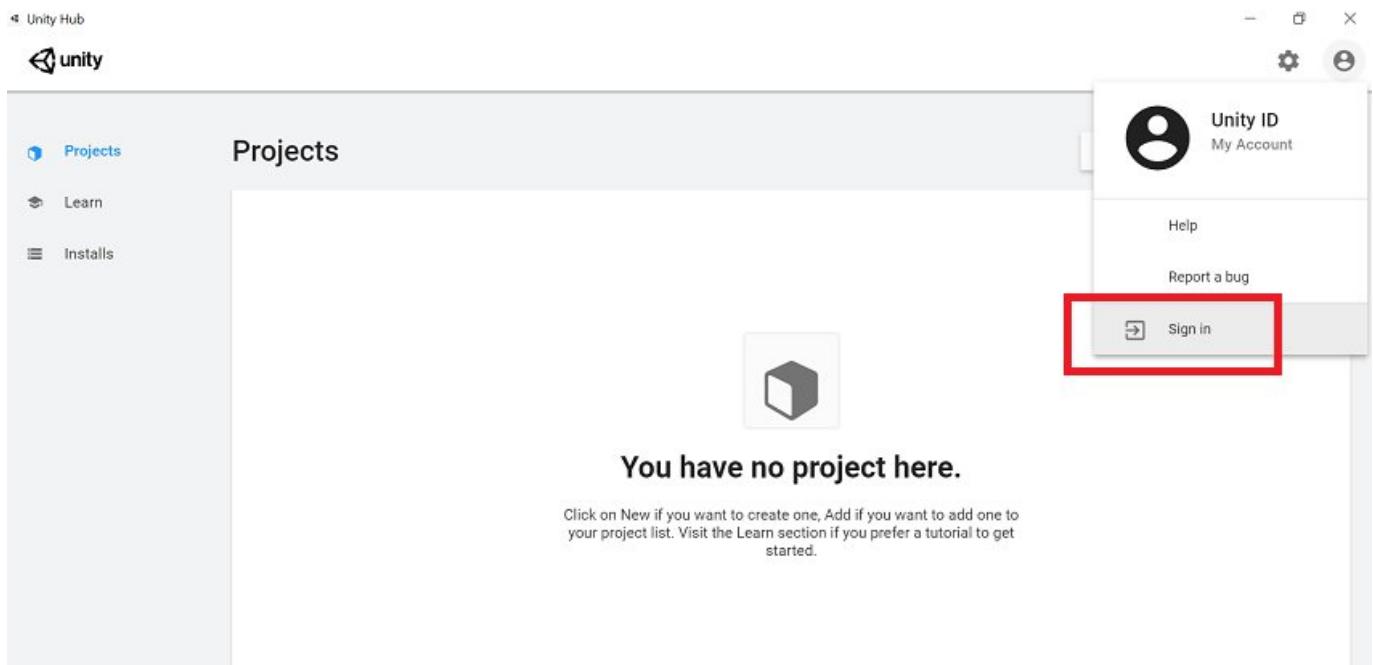
- Now let the installer download and install Unity in your system (PC) and then launch the Unity Game Engine after it gets installed.



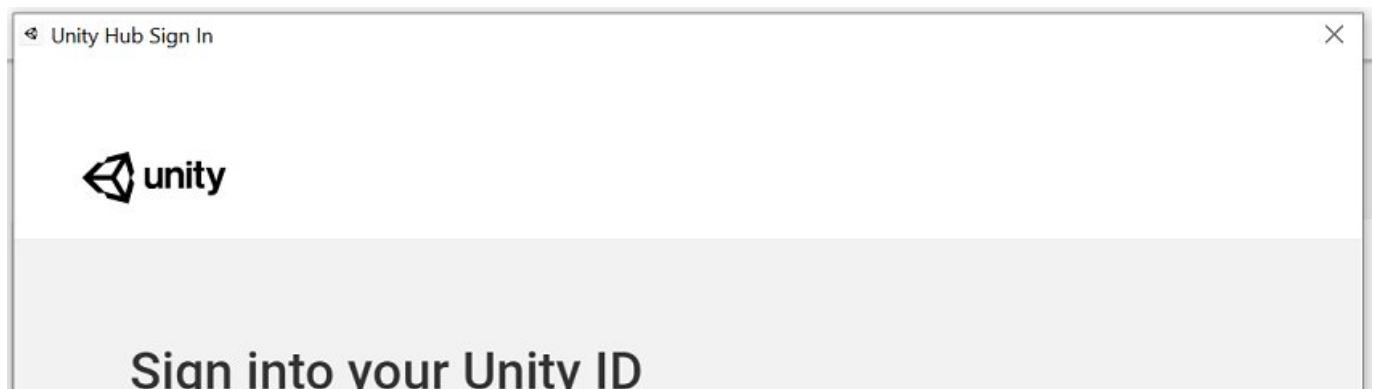


Creating a Unity Account

To use Unity, you have to create an account. For this, open the Unity and go to the top left corner of the window and select the "Sign in" option.



- o Click on the "create one" link.



If you don't have a Unity ID, please [create one.](#)

Email

Or

Sign in with google 

Password

Sign in with facebook 

[Forgot your password?](#)
[Can't find your confirmation email?](#)

[Skip](#) [Sign in](#)

- Enter Email Id, password, Username, and full name then click on "Create a Unity Id" button.

Unity Hub Sign In X



Create a Unity ID

If you already have a Unity ID, please [sign in here.](#)

Email

Password

Username

Full Name

I agree to the Unity [Terms of Use](#) and [Privacy Policy](#)

I understand that by checking this box, I am agreeing to receive promotional materials from Unity

Or

Sign in with google 

Sign in with facebook 

[Create a Unity ID](#)

- Once you are done with filling the enteries, you will receive a confirmation email sent to your email account used to sign up for your account. Confirm your email. Go back to the Unity application and click "continue" after confirming your email.

Unity Hub Sign In X



Confirm your Email

Please check your inbox for a confirmation email. Click the link in the email to confirm your email address.
After you confirm click Continue.

[Re-send confirmation email](#)

[Back to Sign in](#)

[Continue](#)

- o Sign in your Unity account.

Unity Hub Sign In

X



Sign into your Unity ID

If you don't have a Unity ID, please [create one](#).

Email



[Sign in with google](#)

Password

Or



[Sign in with facebook](#)

[Forgot your password?](#)

[Can't find your confirmation email?](#)

[Skip](#)

[Sign in](#)

← Prev

Next →

Youtube For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

[Python Turtle tutorial](#)

[Keras tutorial](#)

Python Pillow

Python Turtle

Keras

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Computer Organization

Discrete Mathematics Tutorial

Discrete Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse

Unity GameObjects

The GameObject is the most important thing in the Unity Editor. Every object in your game is a GameObject. This means that everything thinks of to be in your game has to be a GameObject. However, a GameObject can't do anything on its own; you have to give it properties before it can become a character, an environment, or a special effect.

A GameObject is a container; we have to add pieces to the GameObject container to make it into a character, a tree, a light, a sound, or whatever else you would like it to be. Each piece is called a component.

Depending on what kind of object you wish to create, you add different combinations of components to a GameObject. You can compare a GameObject with an empty pan and components with different ingredients that make up your recipe of gameplay. Unity has many different in-built component types, and you can also make your own components using the Unity Scripting API.

Three important points to remember:

1. **GameObjects can contain other GameObjects.** This behavior allows the organizing and parenting of GameObjects that are related to each other. More importantly, changes to parent GameObjects may affect their children ? more on this in just a moment.
2. **Models are converted into GameObjects.** Unity creates GameObjects for the various pieces of your model that you can alter like any other GameObject.
3. **Everything contained in the Hierarchy is a GameObject.** Even things such as lights and cameras are GameObjects. If it is in the Hierarchy, it is a GameObject that's subject to your command.

Creating and Destroying GameObjects

Some games handle a number of objects in the scene, but we can also create and remove the treasures, characters, and other objects during gameplay.

In Unity, we can create a GameObject using the `Instantiate` function, which makes a new copy of an existing object:

```
public GameObject enemy;
```

```
void Start() {
    for (int i = 0; i < 6; i++) {
        Instantiate(enemy);
    }
}
```

Unity can also provide a `Destroy` function that is used to destroy an object after the frame update has finished or optionally after a short time delay:

```
void OnCollisionEnter(Collision otherObj) {
    if (otherObj.gameObject.tag == "Missile") {
        Destroy(gameObject,.5f);
    }
}
```

Note that the `Destroy` function is also used to destroy individual components without affecting the `GameObject` itself. A common mistake is to write something like:

```
Destroy(this);
```

[← Prev](#)[Next →](#)

[Youtube](#) For Videos Join Our Youtube Channel: [Join Now](#)

[Feedback](#)

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

Aptitude

[Logical Reasoning](#)

Reasoning

[Verbal Ability](#)

Verbal Ability

[Interview Questions](#)

Interview Questions

Company
Interview
Questions

Company Questions

Trending Technologies

Artificial
Intelligence

Artificial
Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science
Tutorial

Data Science

Angular 7
Tutorial

Angular 7

Blockchain
Tutorial

Blockchain

Git Tutorial

Git

Machine
Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures
tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer
Network tutorial

Computer Network

Compiler Design
tutorial

Compiler Design

Computer
Organization and
Architecture

Computer
Organization

Discrete
Mathematics
Tutorial

Discrete
Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse

Unity Components

Unity is a component-based system. Unity components are functional pieces of every GameObject. If you don't understand the relationship between components and GameObjects, first read the GameObjects page before going any further.

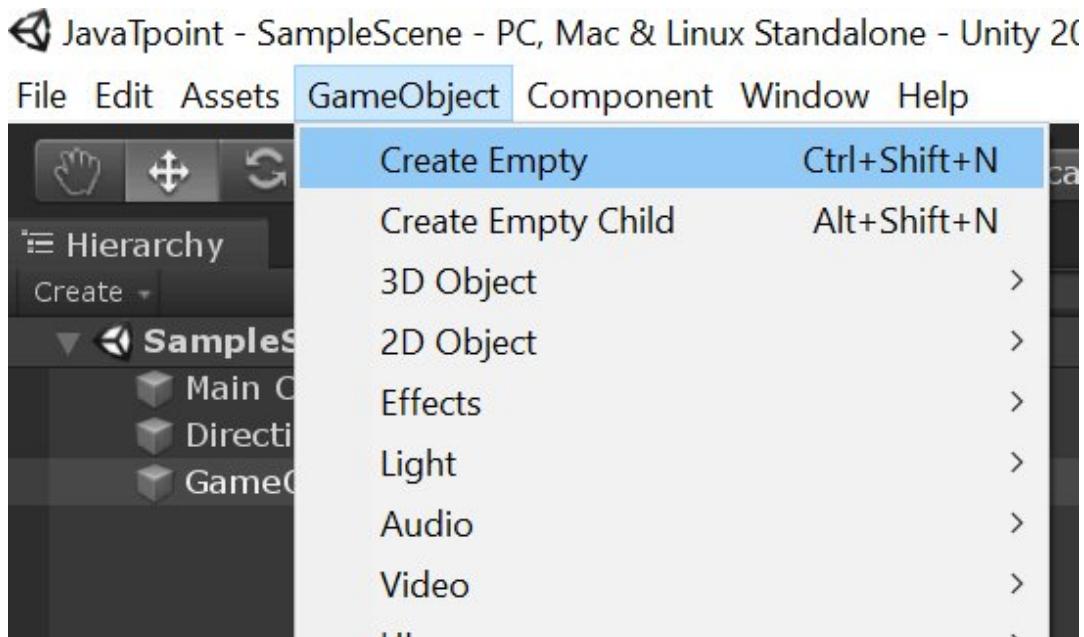
To give functionality to a GameObject, you attach different components to it. Even your scripts are components. So we can say, components are isolated functionality that can be attached to an object to give that functionality to that particular object. That means, when an object requires a specific type of functionality, you add the relevant component.

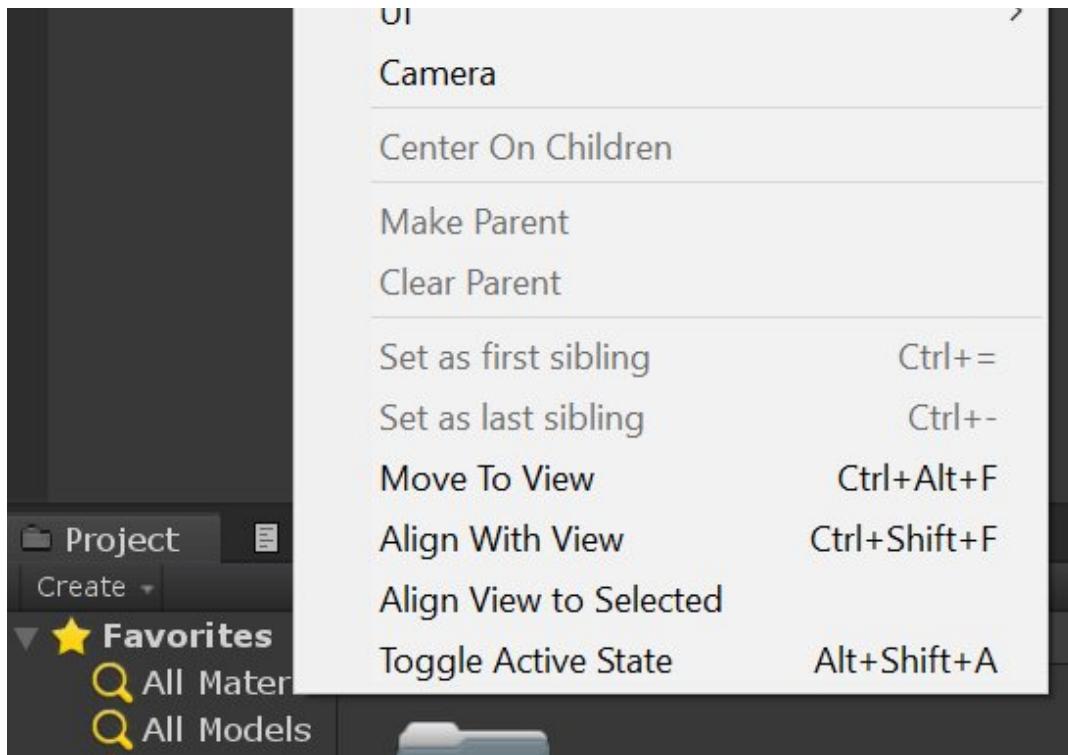
A GameObject is like a container for many different components. By default, all GameObject automatically have a **Transform** component. This is because the Transform defines where the GameObject is located and how it is rotated and scaled. Without a Transform component, the GameObject would not have a location in the world.

Creating a GameObject

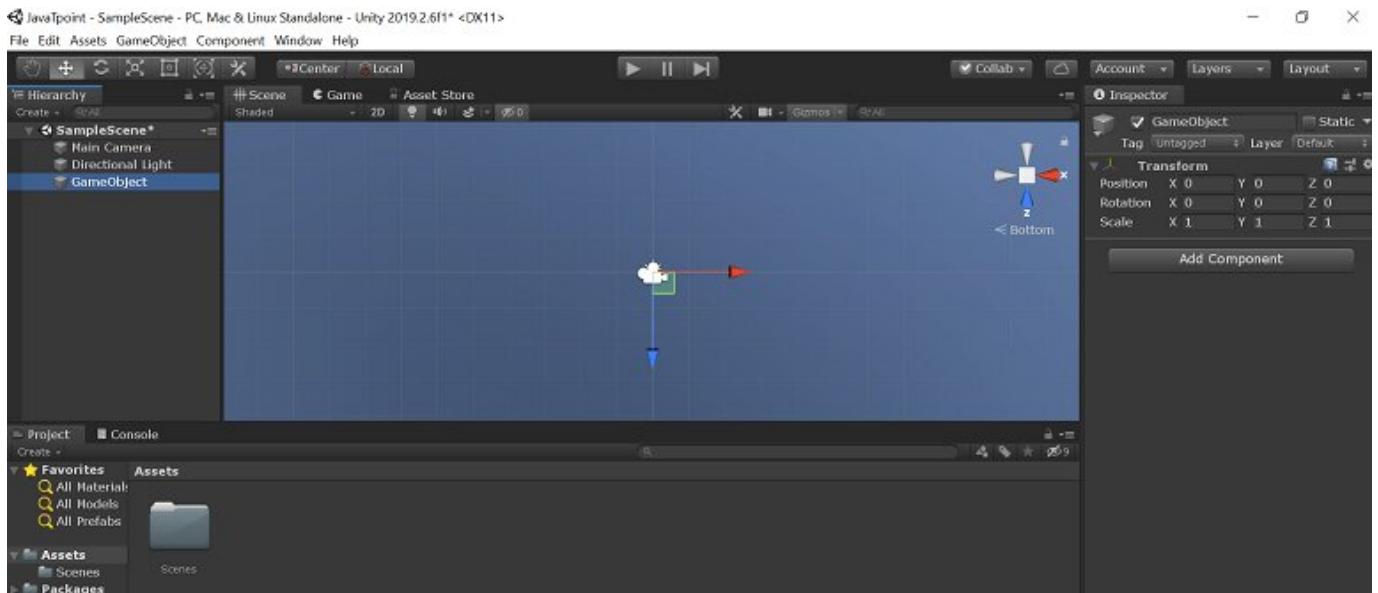
Let's create an empty GameObject:

- From the menu bar click on GameObject -> Create Empty

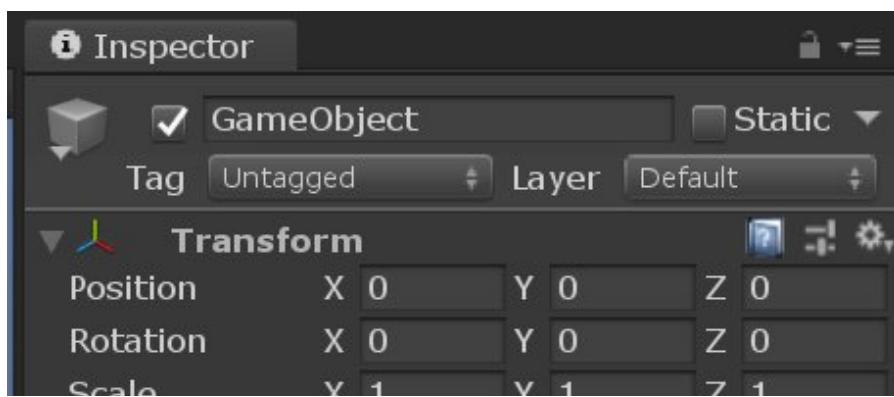


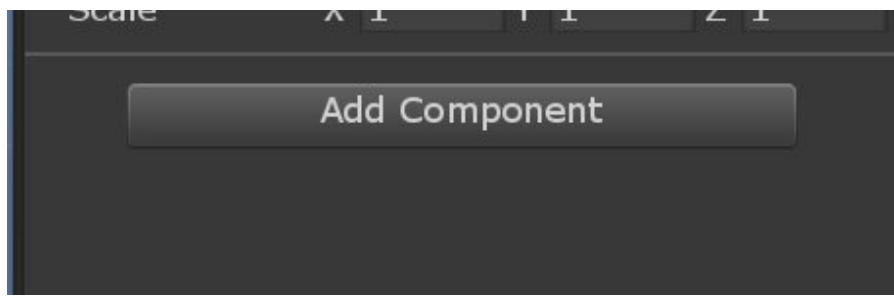


- Select the new GameObject, and look at the **inspector** from the top left corner of the window.



Here we can see that even empty GameObject have a Transform Component.



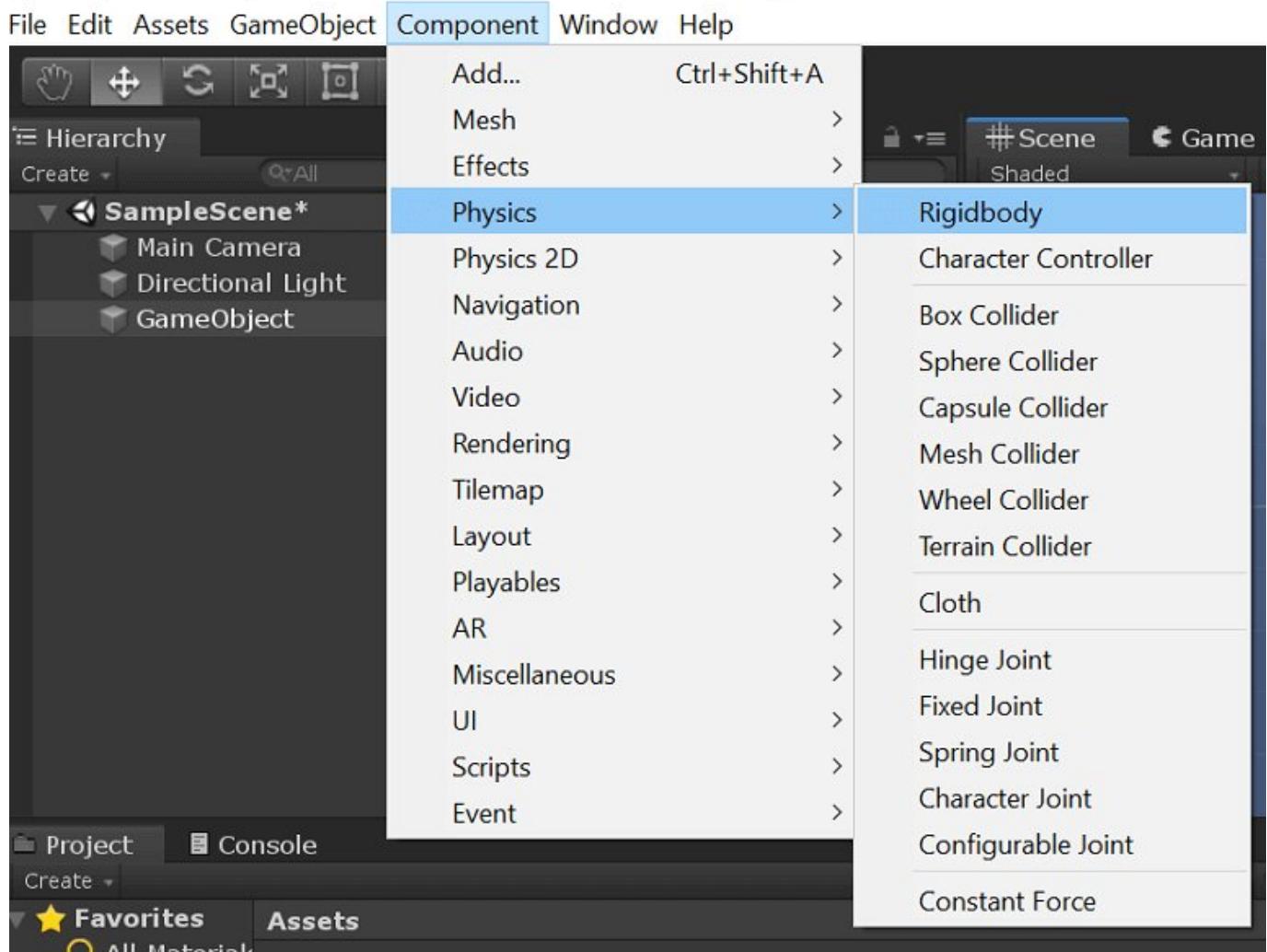


Adding Components

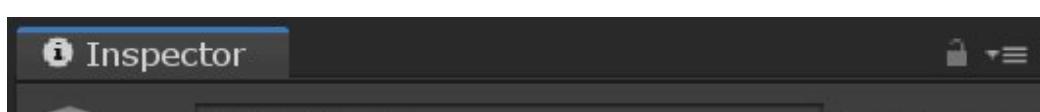
We can add the components to the selected GameObject by the Components menu. Let's try to add a Rigidbody to the empty GameObject we just created. To do that, follow the following steps:

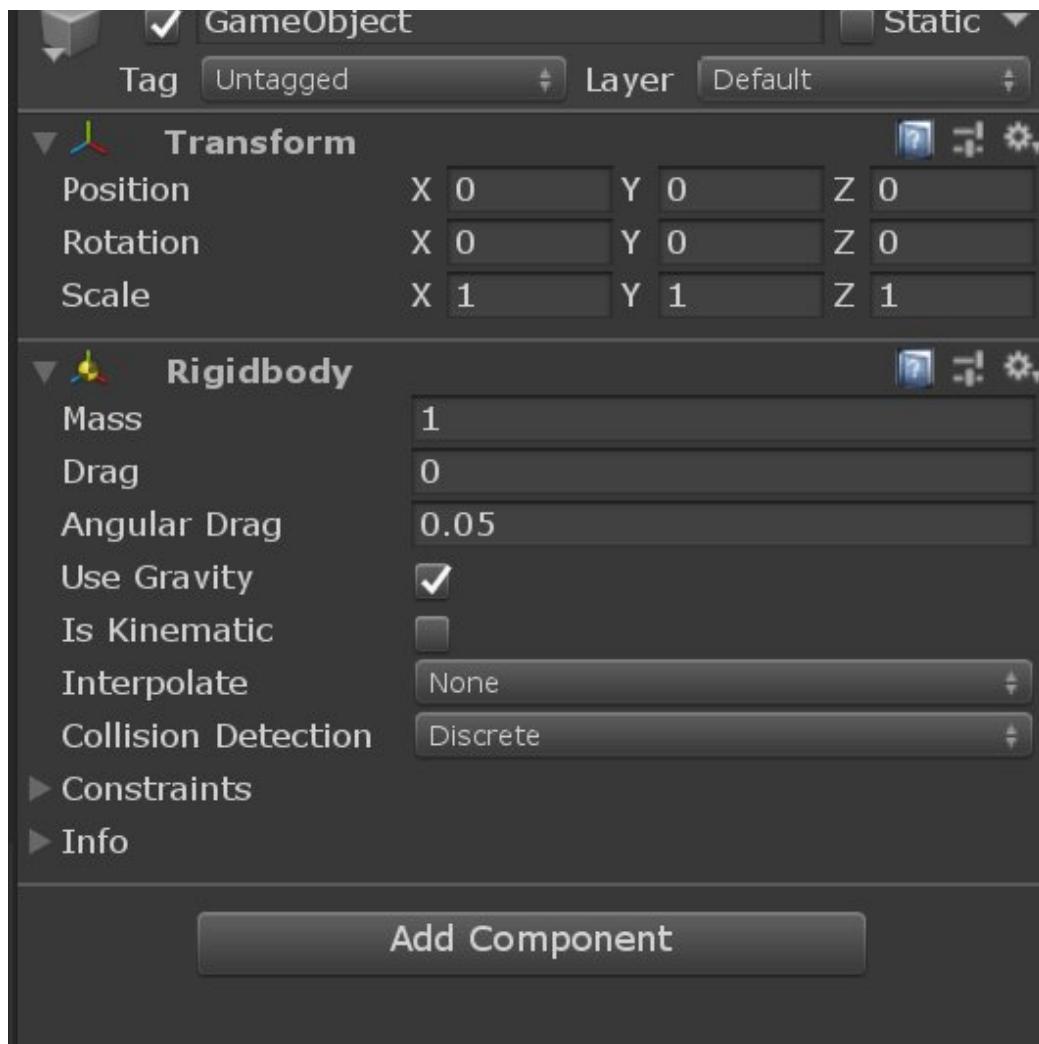
- Select the GameObject and from the menu bar choose Component -> Physics -> Rigidbody.

JavaPoint - SampleScene - PC, Mac & Linux Standalone - Unity 2019.2.6f1* <DX11>

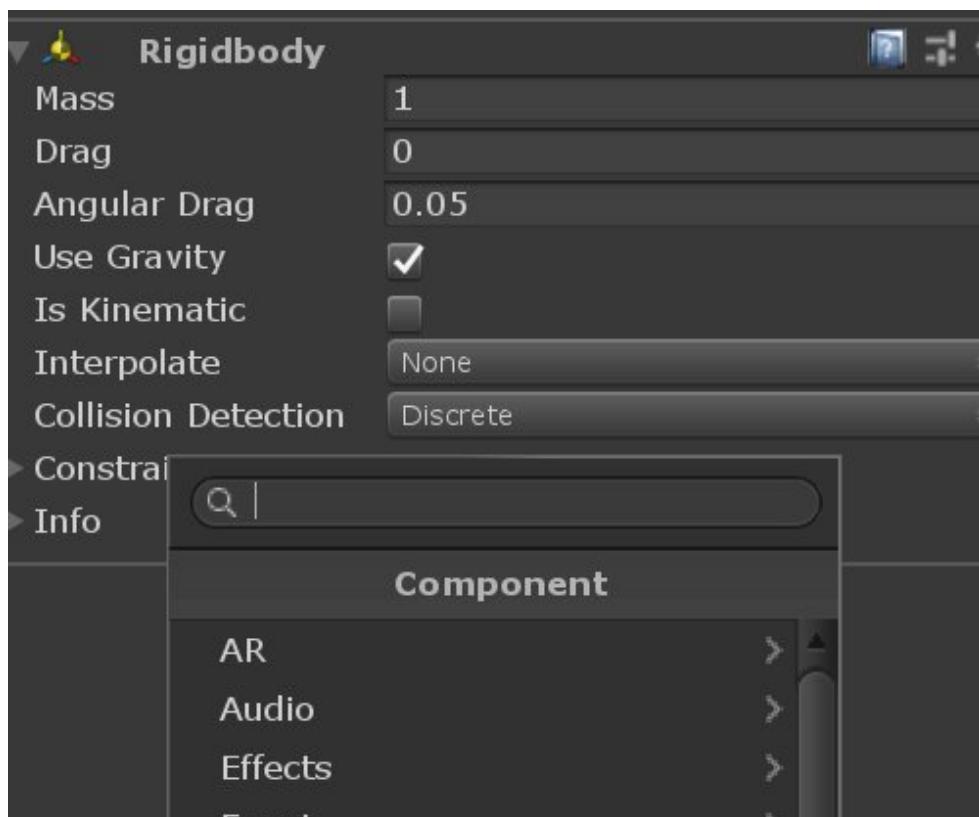


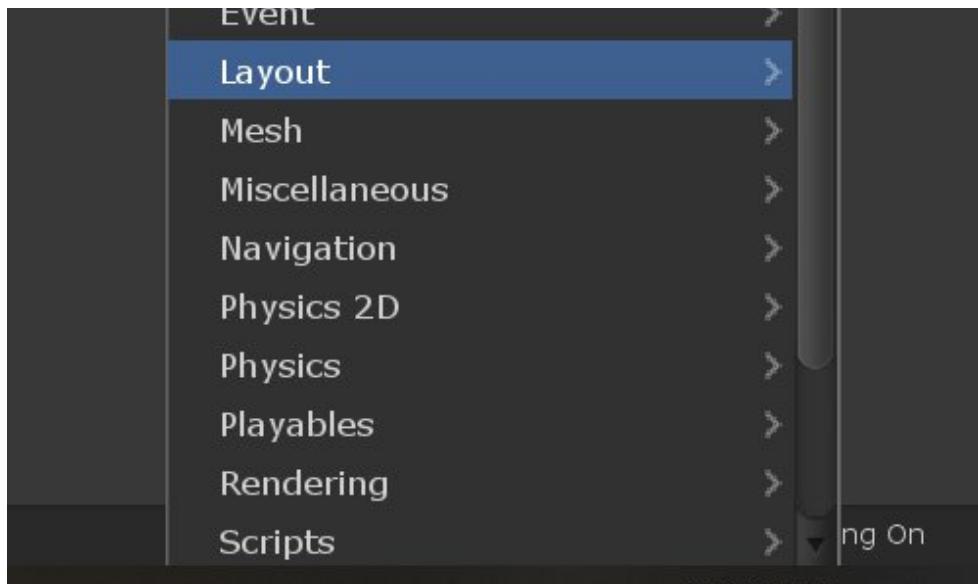
- When you do this, you will get the Rigidbody's properties appear in the inspector.





- Another option is to use the Component Browser to open this Component Browser press 'Add Component' button.





This browser lets you navigate the components by category and also has a search box that you can use to locate components by name.

We can attach any number or combination of components to a single GameObject. One another important feature of Components is flexibility. When we add a component to a GameObject, there are different values of Properties in the component that can be adjusted in the editor while building a game, or by scripts when running the game.

← Prev

Next →

Youtube For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

Splunk tutorial

Splunk

SPSS tutorial

SPSS

Swagger tutorial

Swagger

T-SQL tutorial

Transact-SQL

Tumblr tutorial

Tumblr

React tutorial

ReactJS

Regex tutorial

Regex

Reinforcement learning tutorial

Reinforcement Learning

R Programming tutorial

R Programming

RxJS tutorial

RxJS

React Native tutorial

React Native

Python Design Patterns

Python Design Patterns

Python Pillow tutorial

Python Pillow

Python Turtle tutorial

Python Turtle

Keras tutorial

Keras

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial

AWS Tutorial

Selenium tutorial

Cloud Computing

Intelligence	AWS	Selenium	Cloud Computing
Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular 7 Tutorial
Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial
Cyber Security	Automata	C Programming	C++
Java tutorial	.Net Framework	Python tutorial	List of Programs

Java

tutorial

.Net

Python

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

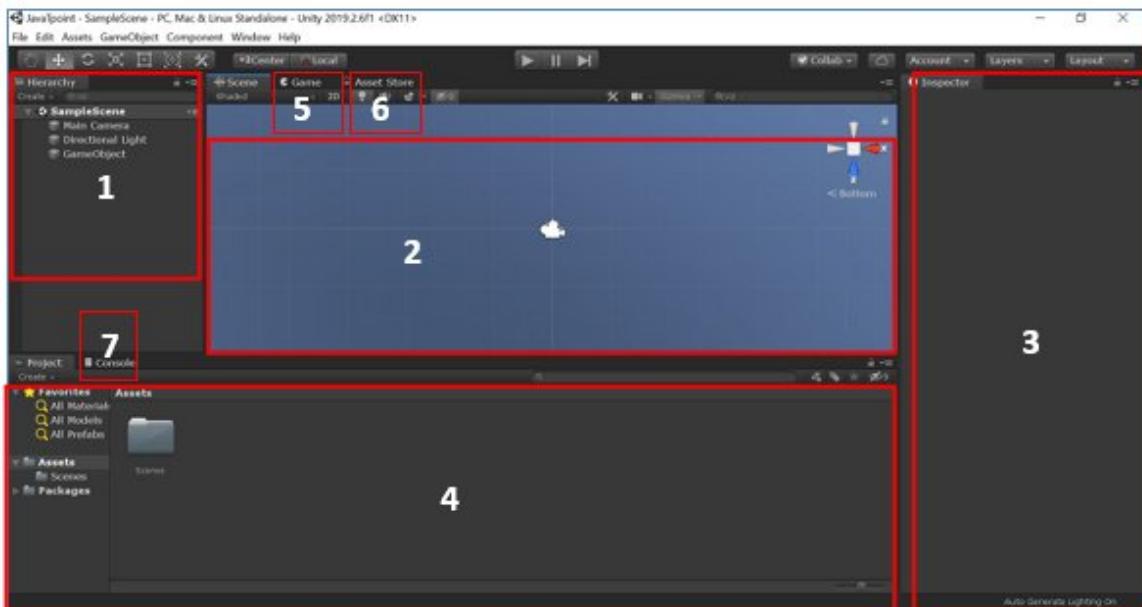
Data Warehouse
Tutorial

Data Warehouse

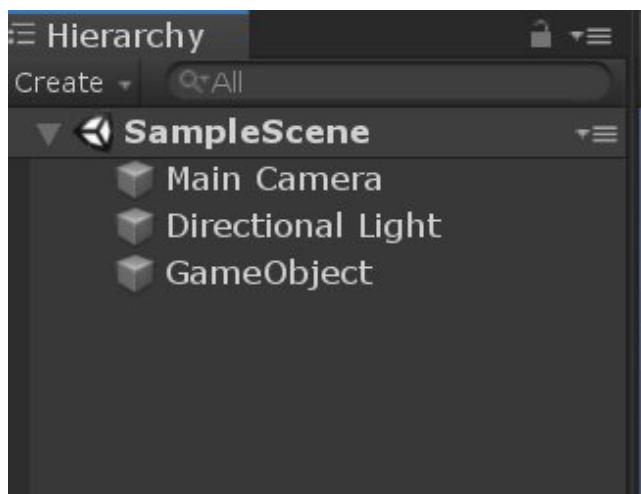
Unity Interface

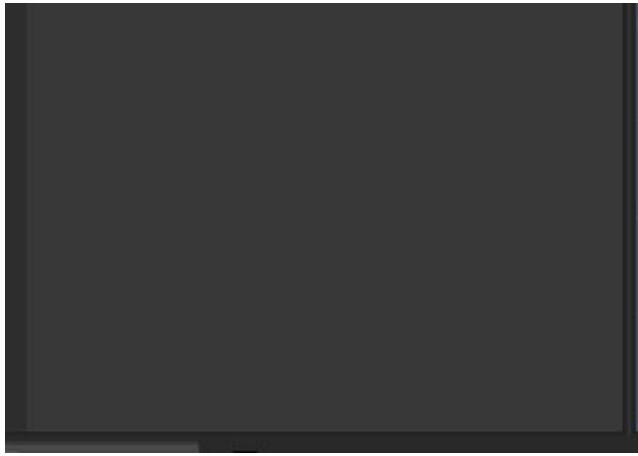
The main editor window is made up of tabbed windows that can be rearranged, detached, grouped, and docked. So we can say that the editor looks different from one project to the next, and one developer to the next, depending on personal preference and what type of work you are doing.

Once your new project is created, and Unity opens, the following window appears:



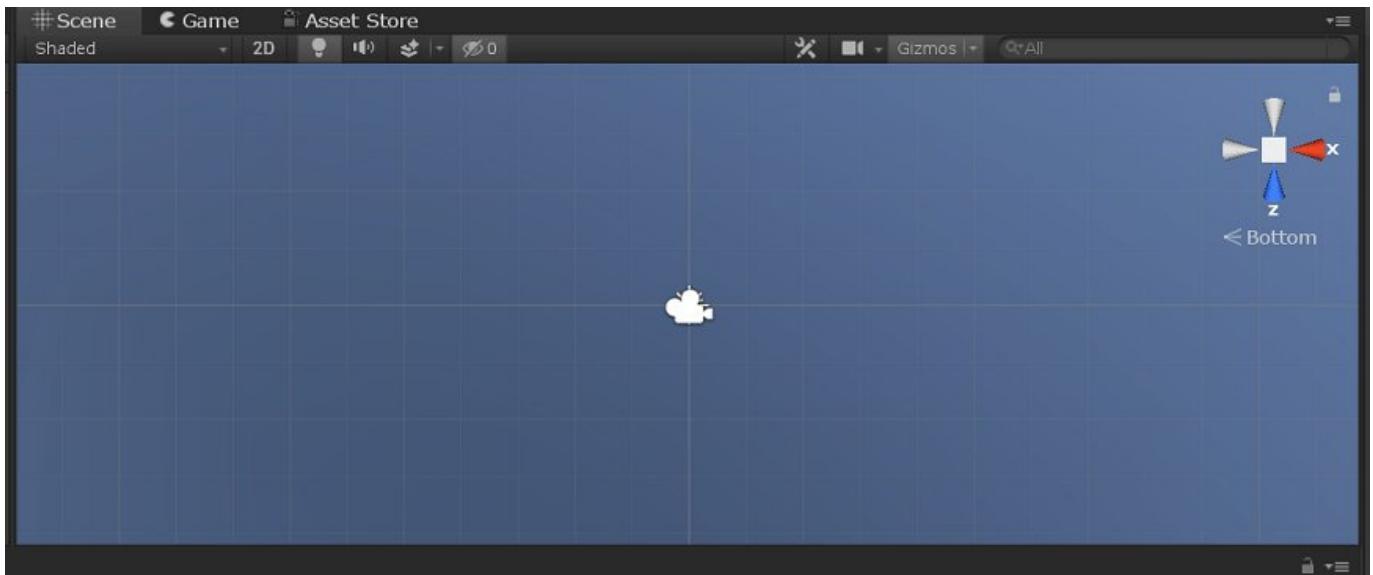
1. Hierarchy Window





- This is the hierarchy window. This is the hierarchical text representation of every object in the scene. It is where all the objects in your recently open scene are listed, along with their parent-child hierarchy.
- Each item in the scene has an entry in the hierarchy, so the two windows are linked. The hierarchy defines the structure of how objects are attached to one another.
- By default, the Hierarchy window lists GameObjects by order of creation, with the most recently created GameObjects at the bottom. We can reorder the GameObjects by dragging them up or down, or by making the parent or child GameObjects.

2. Scene View



- This window is where we will create our scenes. This view allows you to navigate and edit your scene visually.
- The scene view can show a 2D or 3D perspective, depending on the type of project you are working on.
- We are using the scene view to select and position scenery, cameras, characters, lights, and all other types of GameObject.

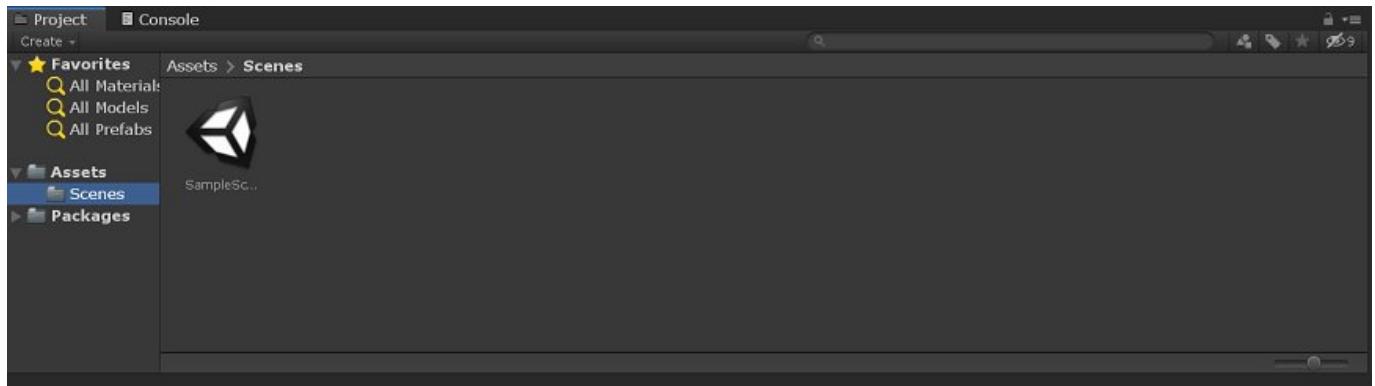
- Being able to select, manipulate, and modify objects in the scene view are some of the most important skills you must learn to begin working in Unity.

3. Inspector Window



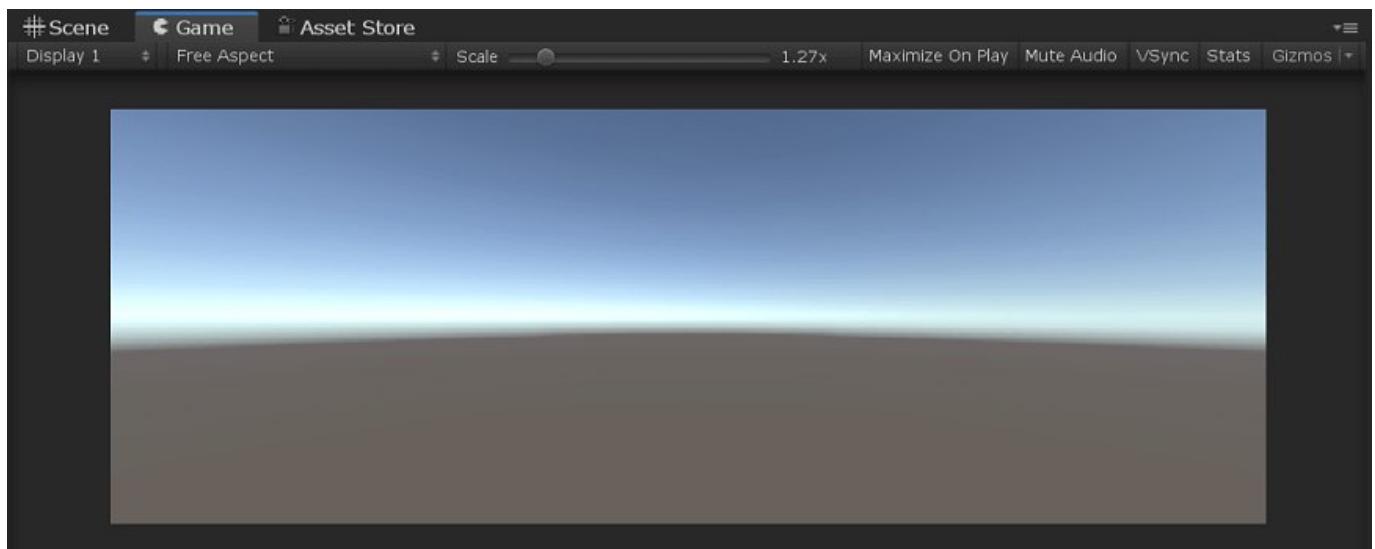
- The Inspector window allows you to view and edit all the properties of the currently selected object.
- Since different types of objects have different sets of properties, the layout and contents of the inspector window will vary.
- In this window, you can customize aspects of each element that is in the scene.
- You can select an object in the Hierarchy window or double click on an object in the scene window to show its attributes in the inspector panel.
- The inspector window displays detailed information about the currently selected GameObject, including all attached components and their properties, and allows you to modify the functionality of GameObjects in your scene.

4. Project Window



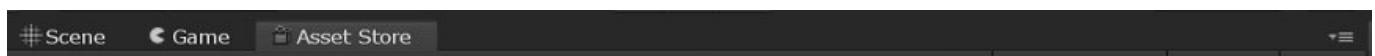
- This window displays the files being used for the game. You can create scripts, folders, etc. by clicking create under the project window.
- In this view, you can access and manage the assets that belong to your project.
- All assets in your project are stored and kept here. All external assets, such as textures, fonts, and sound files, are also kept here before they are used in a scene.
- The favorites section is available above the project structure list. Where you can maintain frequently used items for easy access. You can drag items from the list of project structure to the Favorites and also save search queries there.

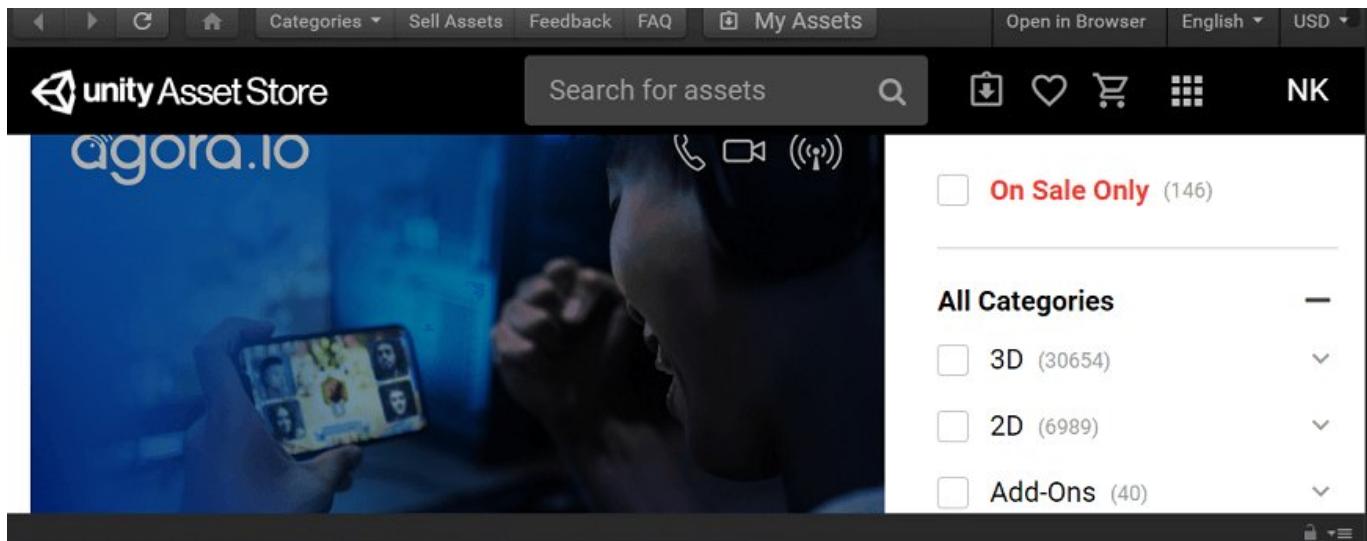
5. Game Window



- This window shows the view that the main camera sees when the game is playing. Means here, you can see a preview window of how the game looks like to the player.
- It is representative of your final game. You will have to use one or more cameras to control what the player actually sees when they are playing your game.

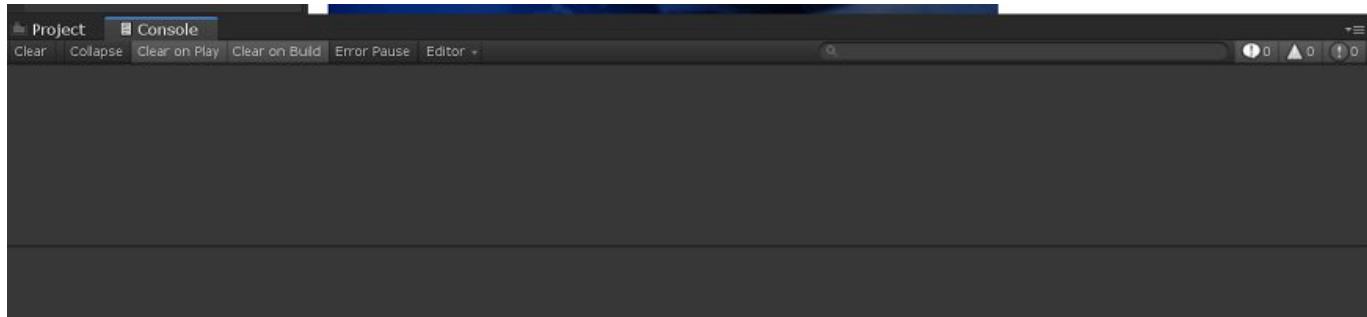
6. Asset Store





- The Unity asset store is a growing library of free and commercial Assets created both by Unity Technologies and also members of the community.
- A wide variety of Assets is available, covering everything from Models, Textures, and animations to whole Project examples, tutorials, and Editor Extensions.
- The assets are accessed from a simple interface created into the Unity Editor and are downloaded and imported directly into your project.

7. Console Window



- If you are familiar with programming, you will already know that all the output messages, errors, warnings, and debug messages are shown here. It is similar for Unity, except output messages are done a bit differently than you think.
- The console window of Unity shows the errors, warnings, and other messages generated by Unity.
- You can also show your own messages in the console using the `Debug.Log`, `Debug.LogError`, and `Debug.LogWarning` function.

← Prev

Next →

Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

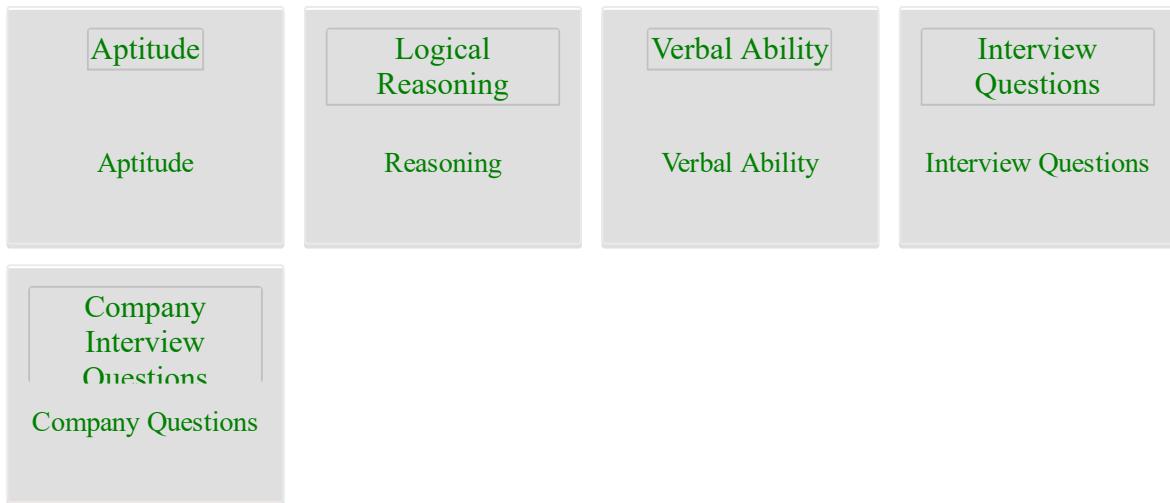
[Python Turtle tutorial](#)

Python Turtle

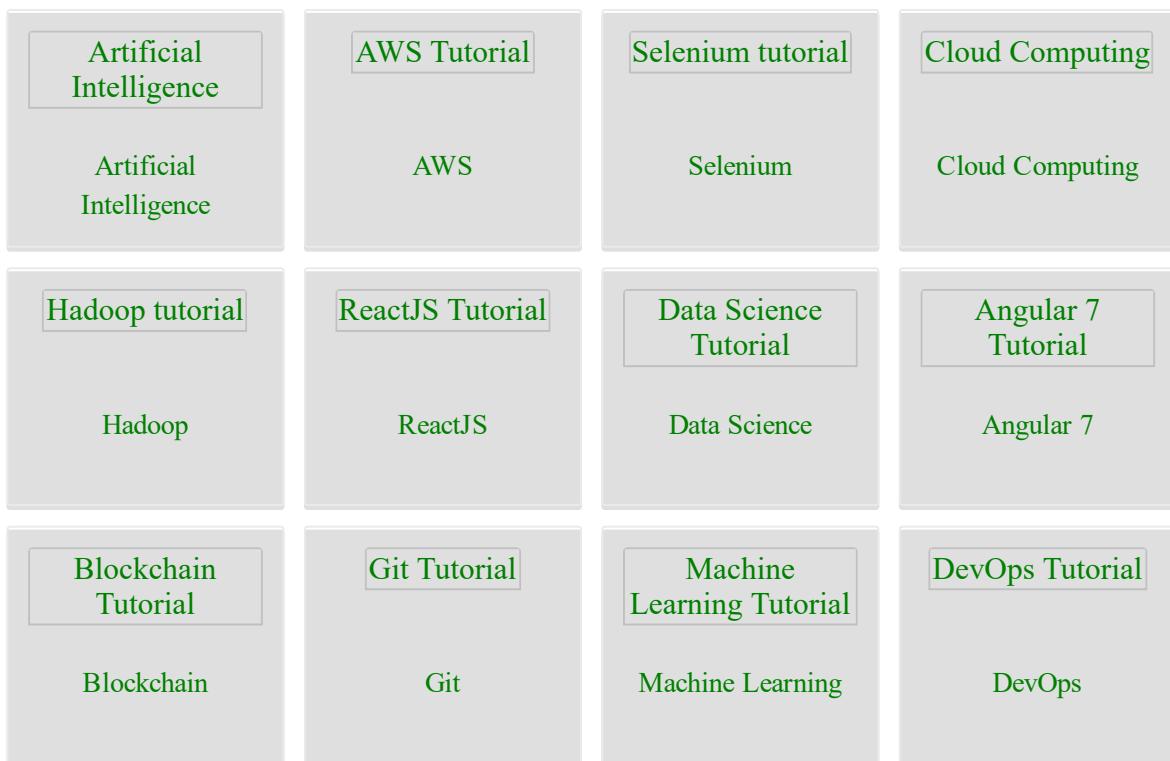
[Keras tutorial](#)

Keras

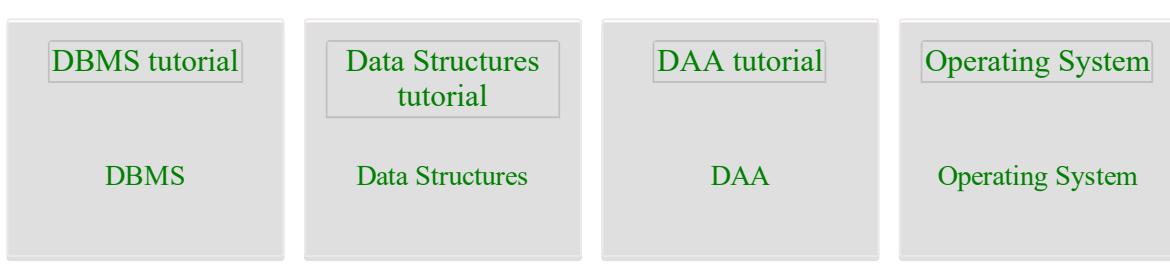
Preparation



Trending Technologies



B.Tech / MCA



Computer
Network tutorial

Computer Network

Compiler Design
tutorial

Compiler Design

Computer
Organization and
Architecture

Computer
Organization

Discrete
Mathematics
Tutorial

Discrete
Mathematics

Ethical Hacking

Ethical Hacking

Computer
Graphics Tutorial

Computer Graphics

Software
Engineering

Software
Engineering

html tutorial

Web Technology

Cyber Security
tutorial

Cyber Security

Automata
Tutorial

Automata

C Language
tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework
tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

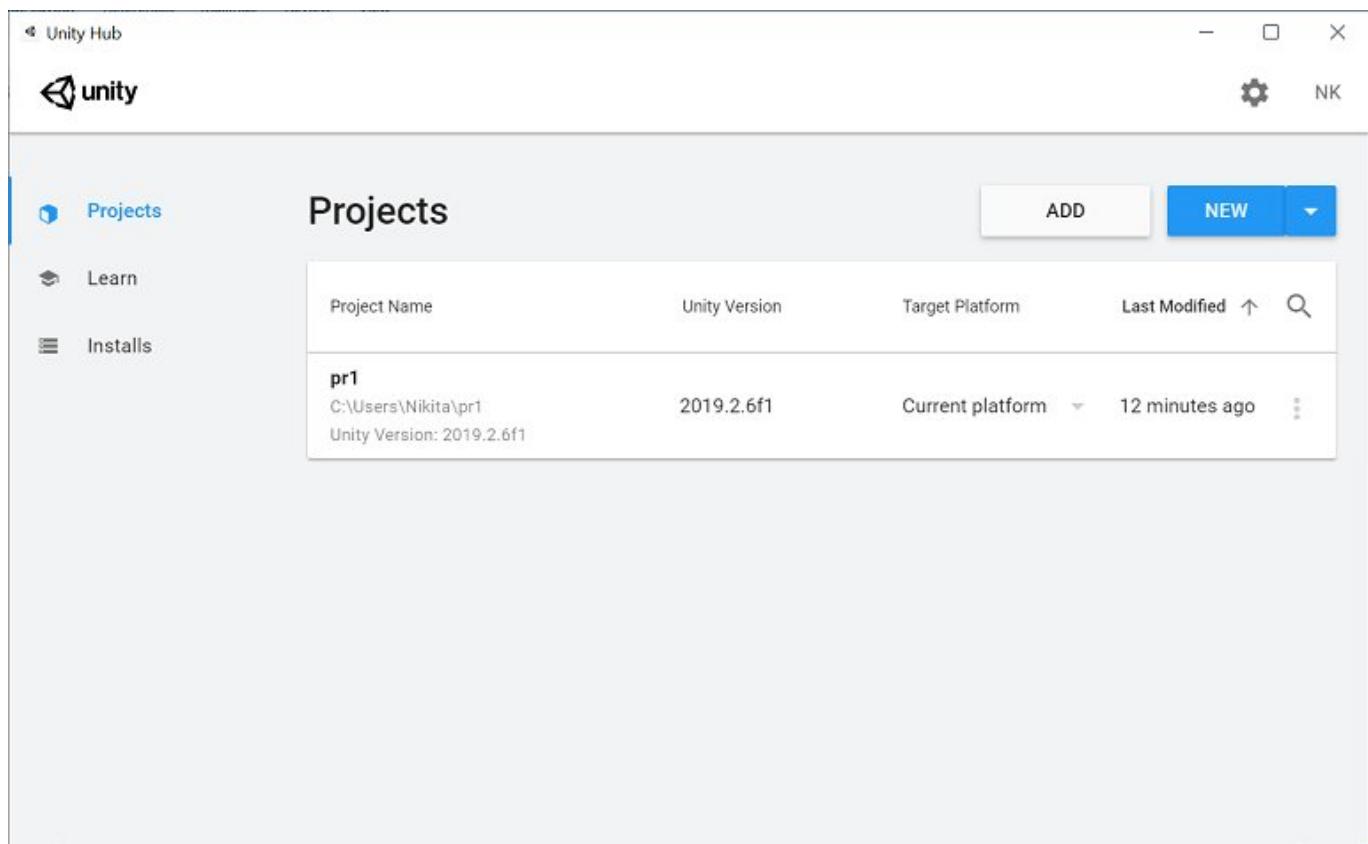
Data Warehouse
Tutorial

Data Warehouse

First Unity Project

To create a new project in Unity, follow the following steps:

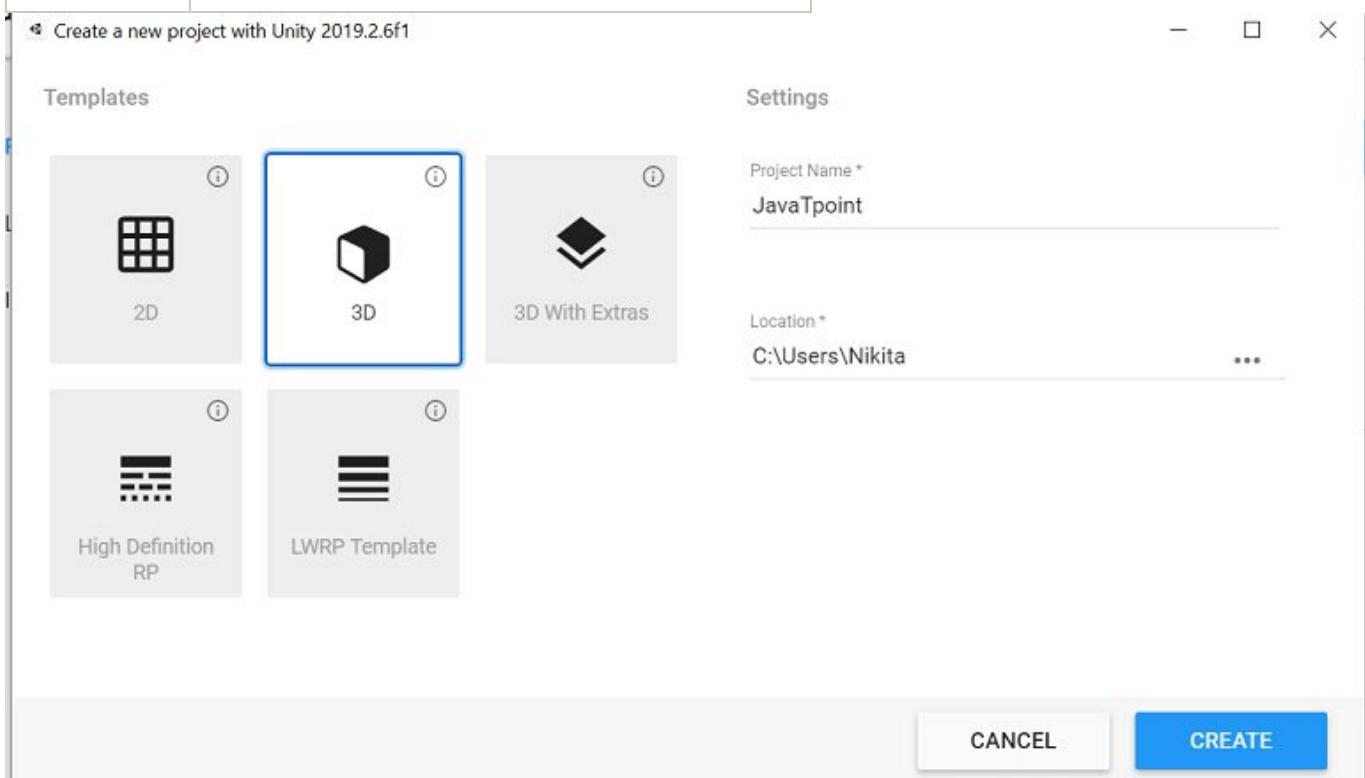
- Open the Unity Hub. In the top right corner of the Home screen, select the **New** button. It will open the 'create a new project with unity' window.



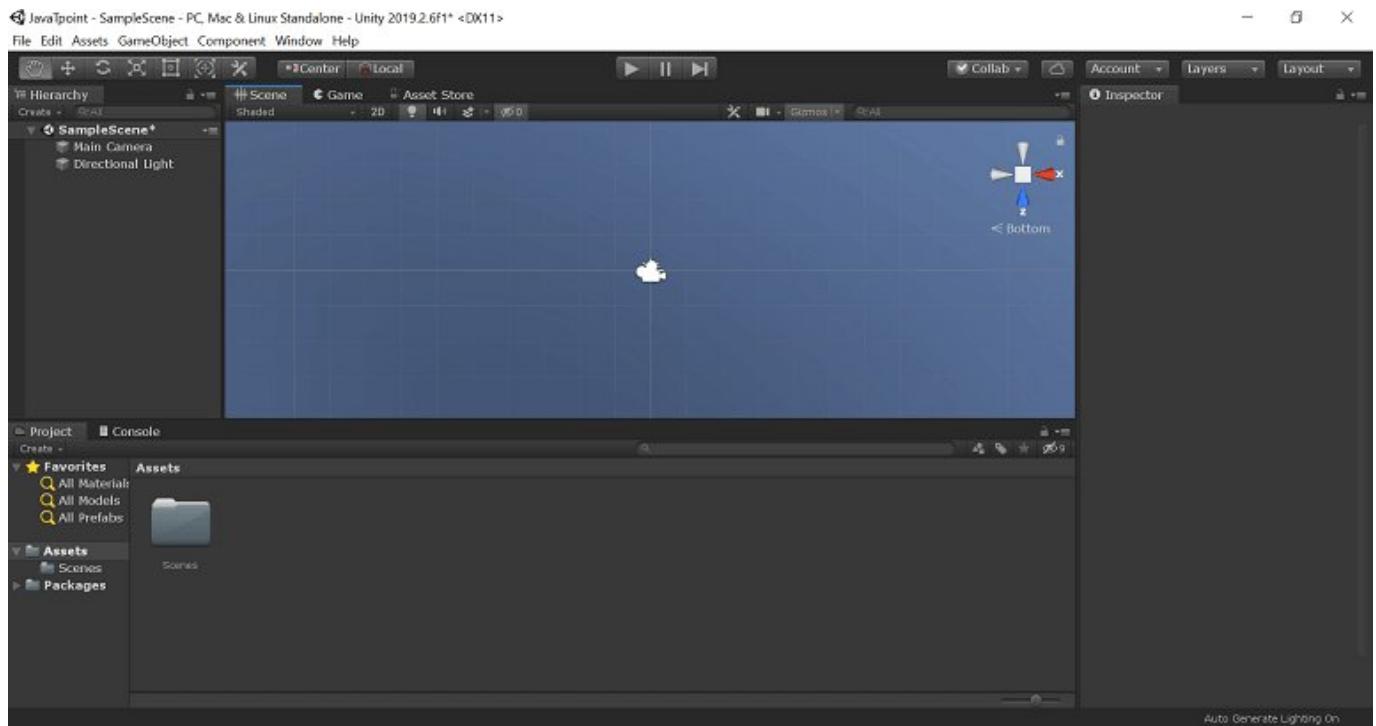
- From the Create Project window, there are various settings for us to apply before Unity creates your project.

Setting	Description
Project name	This is the name of the project you want to give. This name is the folder name that stores Assets, Scenes, and other files related to your project. The default name

	is the New Unity Project; you can change to any name.
Location	This is used to define the location of your project in your computer's files system to store your project. The default location is the home folder of your computer. To change it, enter the file path to your preferred storage location into the Location field. Alternately, click the three blue dots; from here you can navigate to the folder that you want to store your project.
Templates	Select a project template. A project template is used to provide preselected settings based on common best practices for projects. These settings are optimized for 3D and 2D projects across the full range of platforms that Unity supports. The default selected template type is 3D.



- Enter the Create button when you are done. Unity automatically generates essential files and creates your project.



← Prev

Next →

Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

Tumblr tutorial	React tutorial	Regex tutorial	Reinforcement learning tutorial
Tumblr	ReactJS	Regex	Reinforcement Learning
R Programming tutorial	RxJS tutorial	React Native tutorial	Python Design Patterns
R Programming	RxJS	React Native	Python Design Patterns
Python Pillow tutorial	Python Turtle tutorial	Keras tutorial	
Python Pillow	Python Turtle	Keras	

Preparation

Aptitude	Logical Reasoning	Verbal Ability	Interview Questions
Aptitude	Reasoning	Verbal Ability	Interview Questions
Company Interview Questions			Company Questions

Trending Technologies

Artificial Intelligence	AWS Tutorial	Selenium tutorial	Cloud Computing
Artificial Intelligence	AWS	Selenium	Cloud Computing
Machine Learning	ReactJS Tutorial	Data Science	Angular 7

Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular / Tutorial
Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial
Cyber Security	Automata	C Programming	C++
Java tutorial	.Net Framework tutorial	Python tutorial	List of Programs
Java	.Net	Python	Programs
Control Systems	Data Mining	Data Warehouse	

tutorial

Control System

Tutorial

Data Mining

Tutorial

Data Warehouse

Game design and development

**Nature of Games,
Design Elements, Game Components**

lecture 3

Marwa Al-Hadi



link



Zelda

Lecture two part one: Nature of Games



Definitions of Games

- Adams: Fundamentals of Game Design

A **game** is a form of **interactive entertainment** where **players** must overcome **challenges**, by taking actions that are governed by **rules**, in order to meet a **condition**.

- Salen & Zimmerman: Rules of Play

A **game** is a system in which **players** engage in **artificial conflict**, defined by **rules**, that results in a **quantifiable outcome**.



Design Decisions

● Players

- How **many players** are there at a time?
- **Who or what** is the **player** in the **world**?
- Specifies a notion of *identity*

● Goals

- What is the player **trying to achieve**?
- **Defined by** the game or by the player?
- Specifies the player *focus*

Design Decisions

● Rules

- How does the player effect the world?
- How does the player learn the rules?
- Specifies the *boundaries* of the game

● Challenges

- What difficulties must the player overcome?
- Is there more than one way to overcome them?
- Specifies the fundamental *gameplay*

(Other) Design Decisions

● Game Modes

- How are the challenges put together?
- What is the interaction *happend*?

● Setting

- What is the nature of the *game world*?
- What is the *perspective* (e.g. 3D, etc.)?

● Story

- What *narrative or the story* will the player involve?
- How is it **connected** to gameplay?

Play Length

- How short a game can I play and have fun?
 - Least meaningful unit of play
 - **Console**: 30 minutes+ is acceptable
 - **Mobile**: ... think about that
- **Casual** often means **short play units**
 - But can have sophisticated gameplay!
 - **Example**: *Plants vs. Zombies*

Dueling-fighting- Design Philosophies

Narrative-story-

- Games are a *story medium*
 - Focuses on *storytelling*
 - *Traditional narrative structure*
- **Advantages:**
 - Emotionally *compelling*
 - *Strong artistic vision*
- **Disadvantages:**
 - *Author voice* over player voice
 - *Poorly defined mechanics*

Ludic

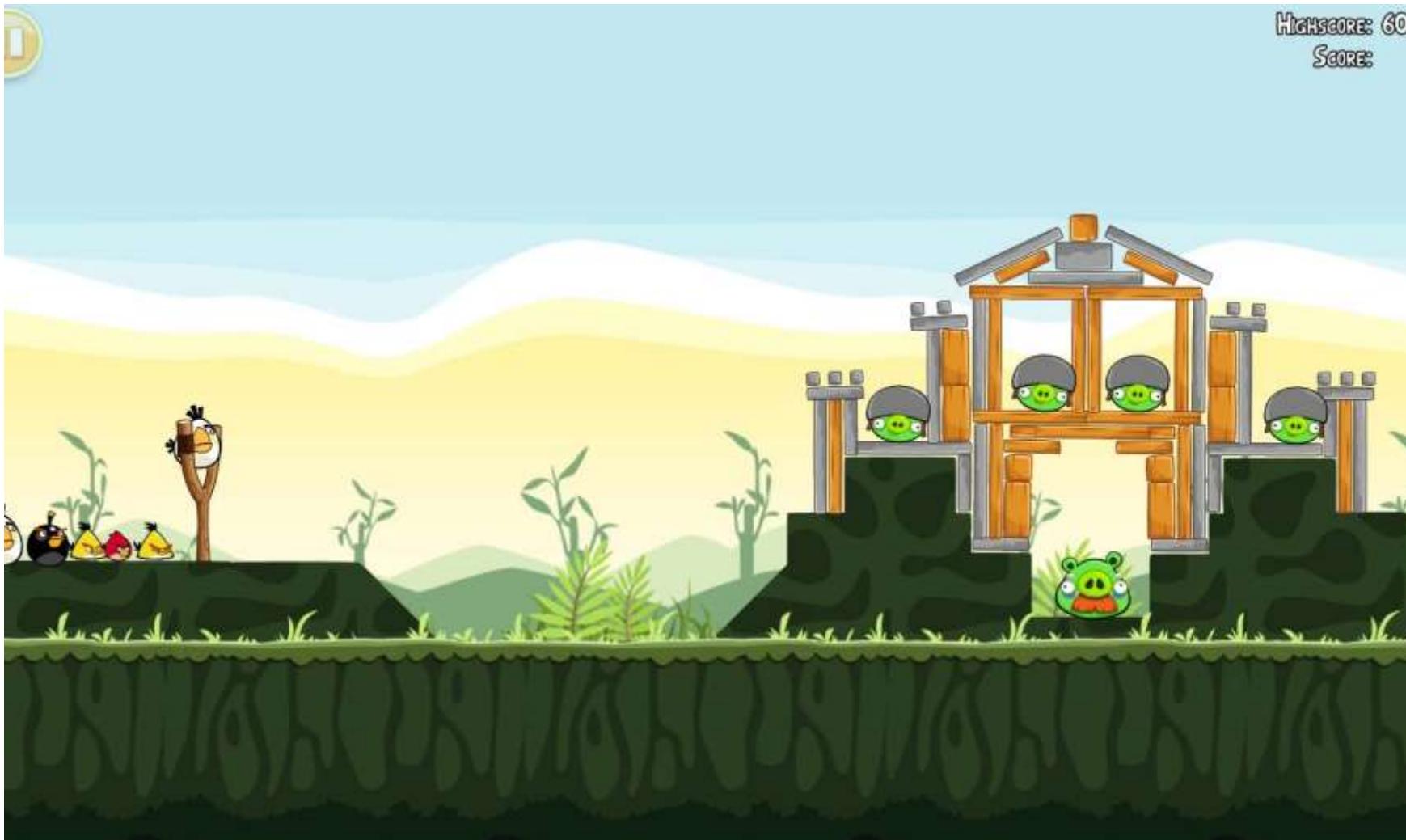
- Games are about *mechanics*
 - Focus on *gameplay, rules*
 - *Storytelling is minimal*
- **Advantages:**
 - Tight, *well-defined gameplay*
- **Disadvantages:**
 - *Lack of player motivation*
 - *Hard to distinguish yourself*

Narrative

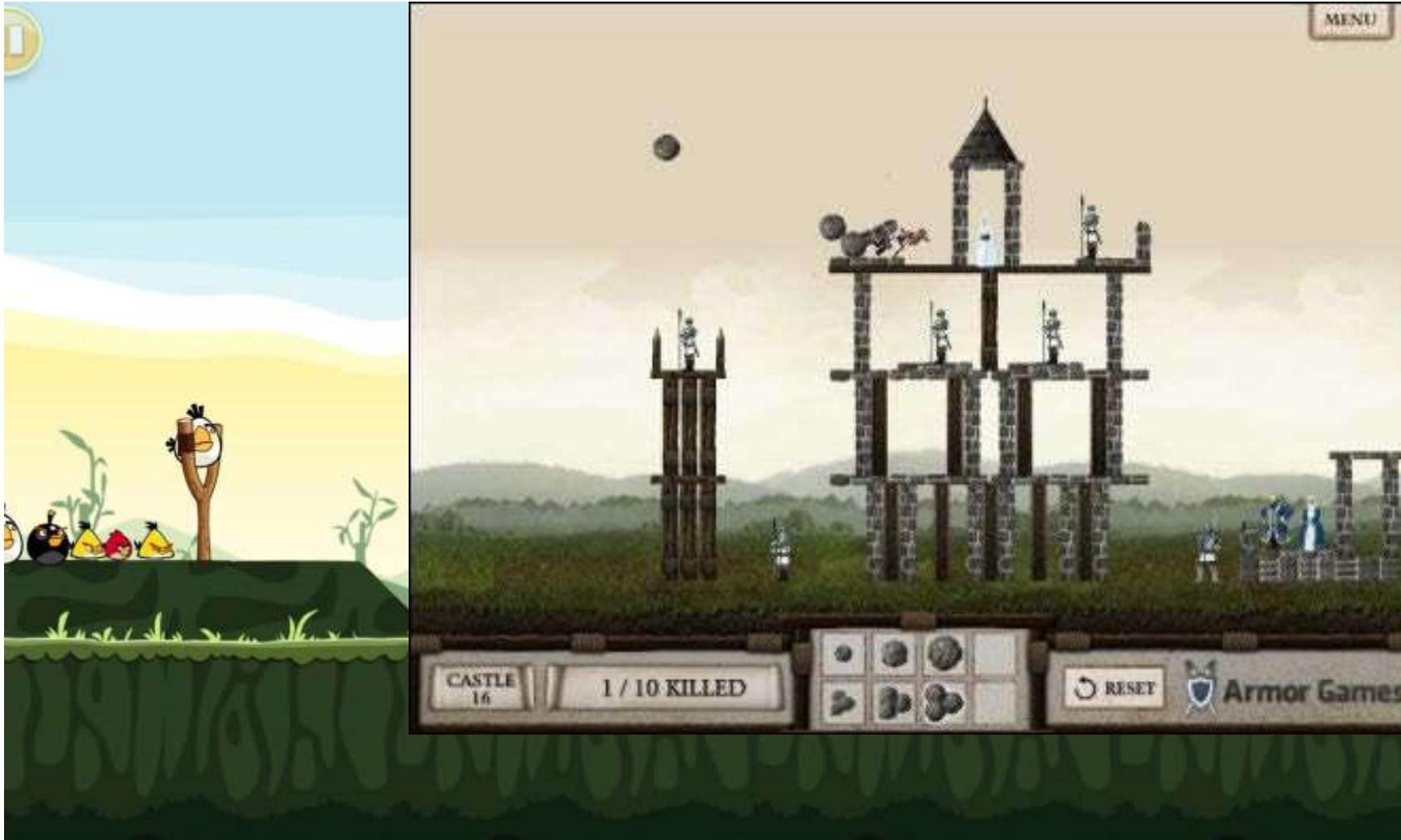
The Dangers of Pure Story



But Ludic is Not Everything



But Ludic is Not Everything



Game Design Must Be a Balance

Motivate the Player

- Needs a story *framework*
 - Setting to work within
 - Strong sense of identity
 - Challenges with context

Empower the Player

- Drama from player *actions*
 - Define what the player can do
 - Challenges reward or punish
 - Freedom in achieving goals

Games are **dramatic**, but they have their own **conventions**.

The Adams Approach

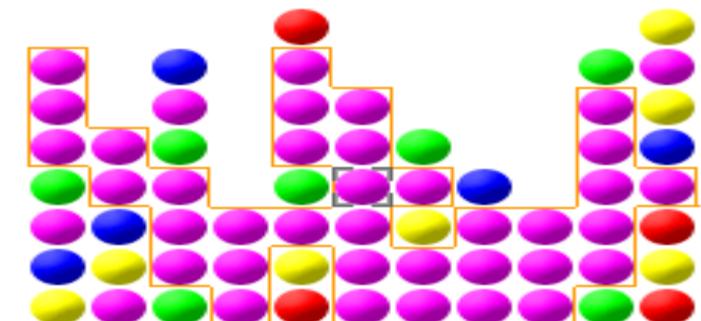
- Games as *wish-fulfillment*
 - I want to..... a game with _
- Questions to answer:
 - What *dream* are you *satisfying* with?
 - What *goals* does this dream create?
 - What *actions* achieve those goals?
 - What *setting* does this dream create?
 - What is the appropriate *interface*?
- Use this to define **gameplay**

Narrative

Ludic

Exploring Gameplay

- **To design games, you must play games!**
 - Experience many different types of gameplay
 - Do not play the same type of game all the time
- **Flash portals are still a good resource**
 - Games are **small** but focus **entirely** on gameplay
 - Ex: Puzzle game



Have Realistic Goals

- **Goal:** Size of a mobile game
 - Can be played instantly with minimal tutorial
- **Quality over Quantity**
 - Ten amazing levels better than 30 poor levels
 - Balance number of challenges with level size

Commercial Examples



- **Braid**: Puzzle platformer with time-travel mechanics
- **Limbo**: Dark platformer with realistic physics
- **Hotline Miami**: Top-down stealth and action
- **Clash of Heroes**: Match 3 + Turn-based strategy
- **Guild of Dungeoneering**: RPG + CCG
 - Use cards to build the dungeon that you explore
- **Monument Valley**: Puzzle-based exploration
- Think about **insdie** games



Examples

- **Mount Sputnick (Spring 2017):**
 - Competitive rock-climbing game
- **Arc en Ciel (Spring 2015):**
 - Platformer where you paint platforms, while enemy erases
- **Dash (Spring 2014):**
 - Action game with dash mechanics to avoid enemies, obstacles
- **Exodus Protocol (Spring 2013):**
 - X-Com style strategy game with only three units
- **Ensembler (Fall 2011):**
 - Classical music rhythm game with you as conductor

Lecture2 part two: Design Elements



Reminder: Aspects of a Game

- **Players:** How do humans affect the game?
- **Goals:** What is the player trying to do?
- **Rules:** How can the player achieve the goal?
- **Challenges:** What obstacles block the goal?

Formal Design Elements

- **Players:** Player Mode Sketches
- **Goals:** Objectives
- **Rules:** Actions and Interactions
- **Challenges:** Obstacles and Opponents

Player Mode Sketches

- Game may have several *player modes*
 - Ways in which player interacts with a game
 - **Example:** Inventory screen
- You should *storyboard* all of your modes
 - Sketches of each of the major player modes
 - May have **action** (like **movie storyboard**)
 - Illustrate how player interacts with game

Dragon Age: Standard Mode



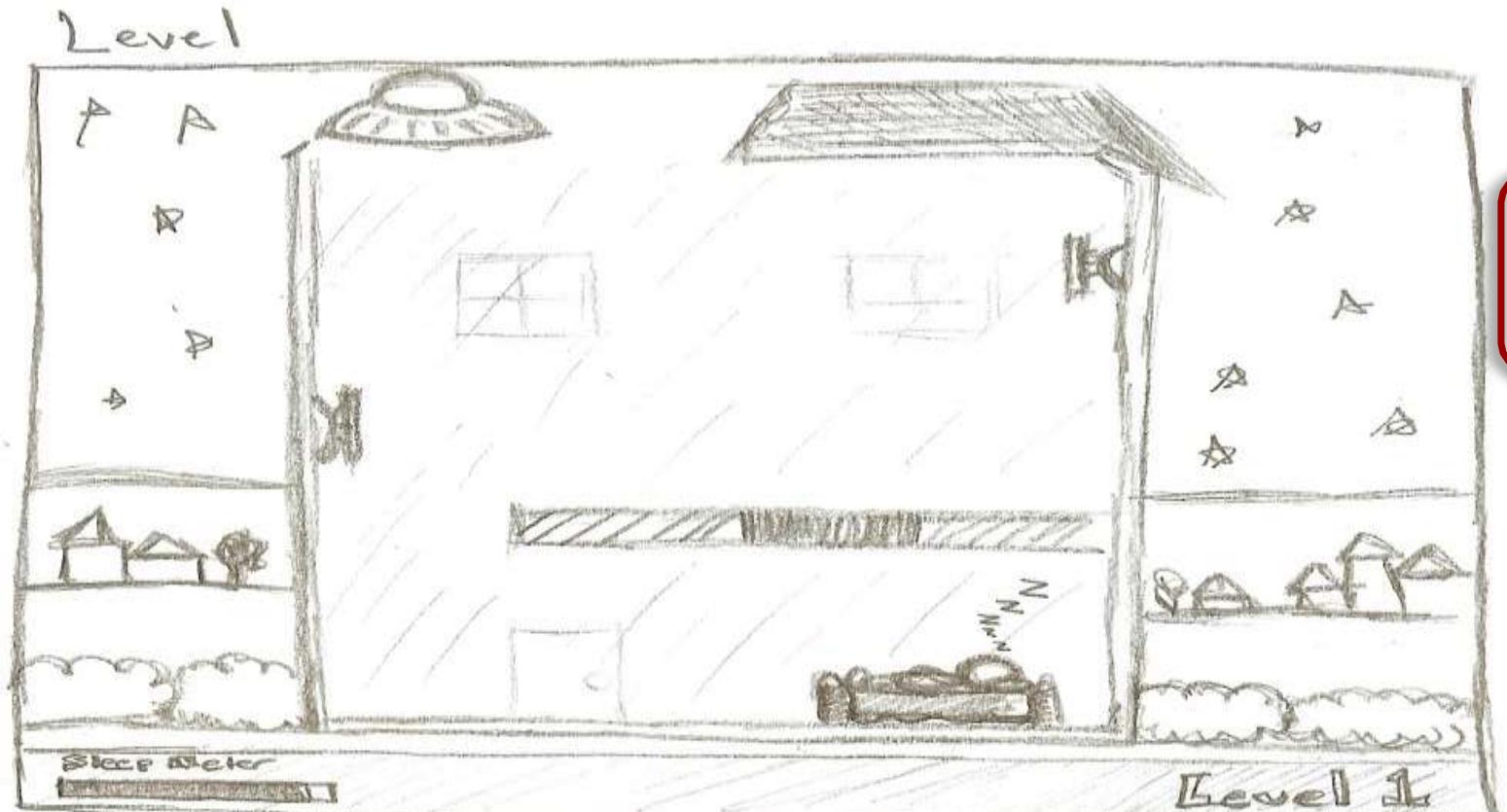
Dragon Age: Inventory-setting mode- Mode



Aside: Help the Hero



Lifted: Player Mode Sketch



also reflected by 2s above head

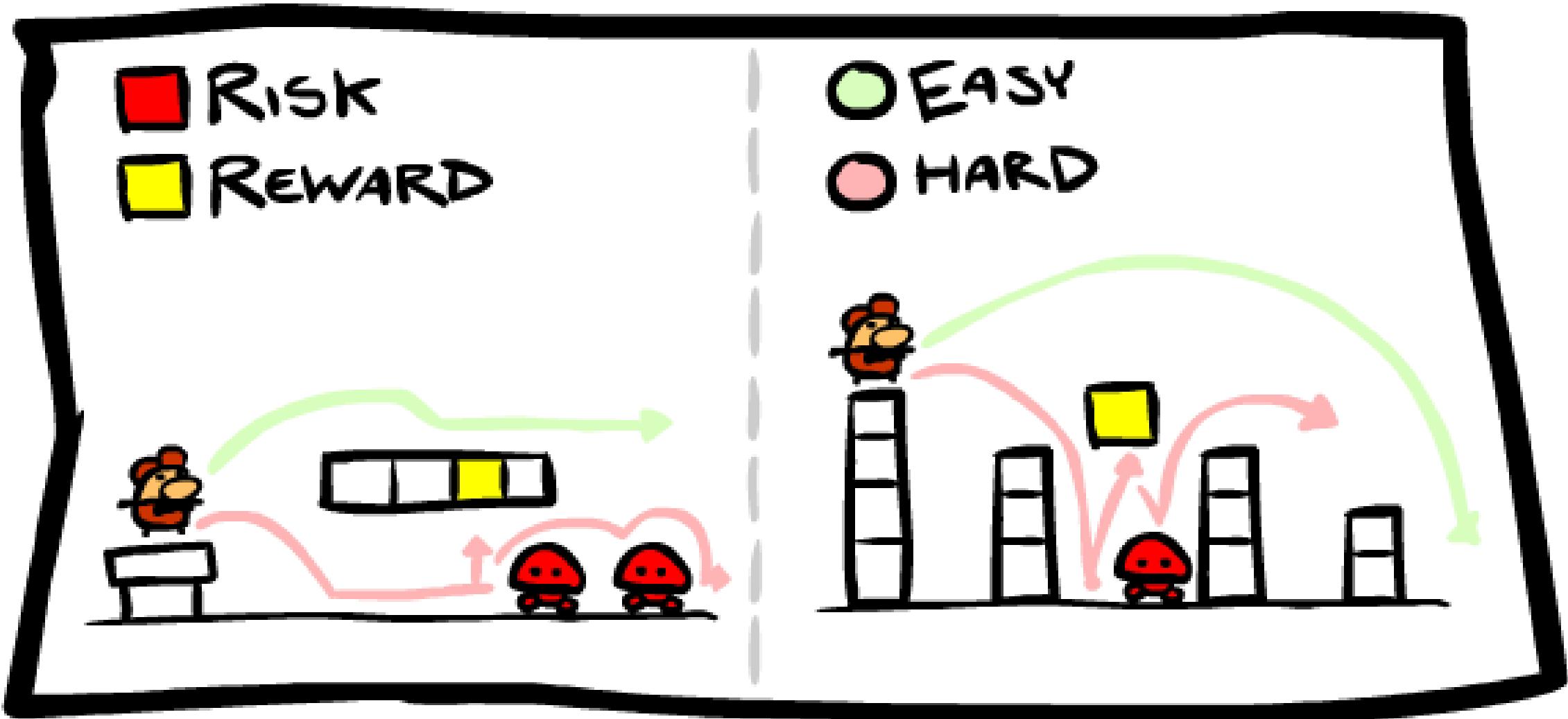
↓ stars
scroll up

Indicating Action

Lifted: Completed Game



Diagramming Action



Objectives

- Anything a player might do their best for
- May be a **primary** game **objective**
 - Progressing the story
 - “Completing” the game
- May be an **auxiliary** game **objective**
 - Side missions/quests
 - Unusual achievements

Objectives

- **Primary** objectives **reflect vision**
 - Help player **realize the dream**
- **Auxiliary** objectives **address player style**
 - Achievements for **achievers**
 - Online resources for **socializers**
- **Player-driven** objectives **require a different focus**
 - Start with a **toy**, and **layer** dramatic elements on it

Some Objective Categories

- **Capture:** take or destroy something of value
 - Includes “kill all enemies of type X”
- **Race:** reach a goal within time
- **Chase:** catch an opponent/enemy
 - Race with a dynamic goal/destination
- **Rescue/Escape:** Get someone to safety
- **Exploration:** Locate something in game world

Actions

- **Verbs that describe what the player can do**

- Walk
 - (left or right) (walk, but faster!)
- Run
 - (up; jump/run for left or right) (left or right)
- Jump
- Shoot

Action
Platformer



Designing Actions

- Starts with **brainstorming the verbs**
 - Define the **types of verbs**
 - Define the **scope of the verbs**
- **Design Goals**
 - **Enough verbs** to avoid being too simple
 - But **not so much** to be **confusing**
 - Do the verbs *directly achieve the goal?*
- Each **verb** maps to a **single input**

Primary Actions



- How **verbs, goals** relate?
 - **Imagine** there **no challenges??!!**
 - What verbs *must* you **have**?
- **Example:** Platformers
 - **Goal:** reach exit location
 - Only need **movement verbs**
 - Killing enemies is *optional*
 - Other actions are *secondary*
- **Focus on primary actions**

Secondary Actions are Optional



- Often in **puzzle platformers**
- Platformer verbs + something
- Directly overcome *challenges*

The Game State

- Collection of values representing **game world**
 - Location, physical attributes of each **game object**
 - Non-spatial values (e.g. health) of these objects
- Actions *modify* the game state
 - Only need **enough state** to understand **interactions**

Interactions

- Not a *direct* action of player
 - Result of the **game state**
 - Can **happen** w/o controller
- **Example: impacts**
 - May be bad (**take damage**)
 - May be good (**power-up**)
- **Other Examples:**
 - Spatial proximity
 - Line-of-sight
 - Resource acquisition



Game Mechanics

● Game mechanic

- Relationship of **verbs**, **interactions**, and **state**
- Often call this **relationship** the “rules”
- **Gameplay** is **display** of these **rules**

● Example: Joust

- **Verbs**: Fly; go left or right
- **Interaction**: fighting with enemies
- **Rule**: If hit enemies, lower player dies

Gameplay Example: Joust



Verbs vs Interactions



- **Design Idea: minimalism**

- Game with very **few verbs**
- Common in **mobile, tablet**

- **Example: Sneak Beat Bandit**

- Has **only one verb: move**
- **Rhythm-balance- game; move to beat**
- All **movement on bars**
- If **obstacle in way, turn**

Avoid Verb Proxies

- **Proxy:** verb that activates another verb

- “Use an item” (what does the item do?)
- “Shoot” (what does the weapon do?)

- **Make the outcome of your verbs clear**

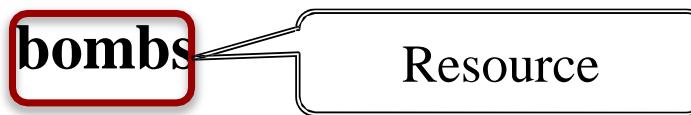
- Fire continuous beam (effects are instantaneous)

- **Important questions to ask**

- How does help reach the goal?
- How is it outcome challenged?



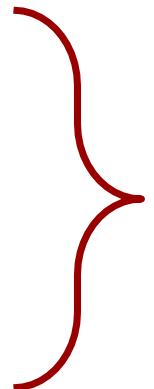
Challenges: Limitations

- You **cannot** always **perform** an **action**
 - Shooting may require **bombs** 
 - Cannot (always) jump in mid air
- **Limitation:** requirement to perform action
 - Boolean test (like an **if-then**)
 - Checked at time of user input
- Only **one limitation** per verb
 - If more than one, **split into more verbs**
 - Reason **double-jump is distinct**

Challenges: Resources

- Resources are **non-spatial** part of game state
 - Any value **not a location or physical attribute**
 - May be global or **attached to an entity**
- Examples
 - **Entity:** bombs, health points
 - **Global:** enemy creating, time remaining
- They also **define the game economy**

Putting It All Together

- Start with your **vision**
 - This creates **setting** and **player goals**
 - Create a (partial) list of the following:
 - **Objectives**
 - **Actions**
 - **Interactions**
 - **Challenges**
- 
- Sketch **player modes** to show them in **action**

Lecture2 part three: Game Components



Starting Prompt

- What exactly is a **game engine**?
 - What **libraries** does it have to **provide**?
 - What **tools** **need** to **come** with it?
- What **skills** should an **engine** require?
 - **Extensive programming** experience ?
 - **Minimal programming** experience ?
 - **No programming** experience ?
 - **Artistic ability** (vs. paying for assets)?



So You Want to Make a Game?

- Will **assume** you have a *design document*
 - Focus of next week and a half...
 - Building off the **ideas** of previous work
 - Need to **assign tasks** to the **team** members
 - Helps to **break game** into *components*
 - Each **component** being a **logical unit** of work.



Traditional Way to Break Up a Game

- **Game Engine**

- Software, created primarily **by programmers**

- **Rules and Mechanics**

- Created **by the designers**, with programmer input

- **User Interface**

- Coordinated with **programmer/artist/HCI specialist**

- **Content and Challenges**

- Created primarily by **designers**



Features of Game Engines

- Power the **graphics** and **sound**
 - **3D** rendering or **2D** sprites
- Power the character and strategic **AI**
 - Typically **custom designed** for the game
- Power the **physics interactions**
 - Must support impacts at a simple minimum
- Describe the **systems**
 - **Space** of possibilities in game world



Commercial Game Engines

- Libraries that take care of **technical tasks**
 - But *systems* always need some specialized code
 - Game studios buy *source code licenses*
- Is **LibGDX** a game engine?
 - It has **libraries** for **graphics, physics, and AI**
 - But you **still have to provide code for systems**
- **Basic bones engine: graphics, physics, audio**

Game Engines: Graphics

- Minimum requirements:

1. API to import artistic assets([interface](#))
2. Routines for manipulating images



- Two standard 3D graphics APIs

- **OpenGL**: Unix, Linux, Macintosh
- **Direct3D**: Windows



Game Engines: Physics

- Defines **physical attributes** of the world

1. There is a **gravitational** force
2. Objects may have **contact**
3. Ways in which **light** can **reflect**



- Does **not** define **precise values or effects**
 - The *direction* or *value* of **gravity**
 - Friction *constants* for each object
 - Specific *lighting* for each material

Game Engines: Systems

- Physics is an example of a game **system**
 - Specifies the space *of possibilities* for a game
 - But **not** the *specific parameters* of elements
- Extra code that you add to the **engine**
 - Write **functions** for the **possibilities**
- Programmer vs. *gameplay designer*
 - **Programmer** creates the system
 - **Gameplay designer** fills in parameters

Systems: Super Mario Bros.

● Levels

- **Fixed height** scrolling maps
- Populated by **blocks** and **enemies**

● Enemies

- Affected by **stomping** or **bumping**
- Different movement/AI schemes
- Spawn **enemies**

● Blocks

- Can be **stepped on safely**
- Can be **bumped** from **below**

● Mario (and Luigi) can be **small, big, or fiery**



Characteristics of an Engine

- Broad, adaptable, and extensible
 - ~~Encodes all *non mutable changeable* design decisions~~
 - ~~Parameters for all *mutable changeable* design decisions~~
- Outlines gameplay **possibilities**
 - Cannot be built **independent** of design
 - But only **needs highest level information**
 - **Gameplay specification** is sufficient

Data-Driven Design

- **No code outside engine**; all else is data
 - Create **game** content with **level editors**
- **Examples:**
 - Art, music in industry-standard file formats
 - Object data in JSON or other data file formats
 - Character behavior specified through scripts
- Major focus for alpha release

Popular Indie Engines

56



- Use data-driven design
- **All code is in “scripts”**
- Core code is inaccessible
- Now engines all in-house



Data driven : that all decisions and processes are dictated by the data and decision made base on user track

Rules & Mechanics

- Fills in the values for the system
 - Parameters (e.g. gravity, damage amounts, etc.)
 - Types of player abilities/verbs
 - Types of world interactions
 - Types of obstacles/challenges
- But does not include ~~specific challenges~~
 - Just the list all challenges that ~~could~~ exist
 - Contents of the *palette* for level editor

Rules: Super Mario Bros.



Spinys

● Enemies

- Spinys damage Mario when stomped
- Piranha Plants aim fireballs at Mario



Piranha Plants

● Environment

- Mushroom makes Mario small
- Fire flower makes Mario big and fiery



Game AI: Where Does it Go?

- Game AI is traditionally placed in **mechanics**
 - AI needs rules to make **right choices**
 - AI to give **characters** **personalities**
- But it is **implemented** by programmer
 - Search algorithms/machine learning
- “AI Photoshop” for designers



Interfaces

- Interface specifies
 - How player does things (player-to-computer)
 - How player gets feedback (computer-to-player)
- More than engine+mechanics
 - Describes what the player can do
 - Do not specify how it is done
- Bad interfaces can kill a game

Interface: Dead Space-video-



Designing Visual Feedback

- Designing for **on-screen** activity
 - **Details** are **best processed** at the **center**
 - Peripheral **vision** mostly **detects motion**
 - Visual **highlighting** around **special objects**
- Designing for **off-screen** activity
 - **Flash** the **screen** for **quick events** (e.g. being hit)
 - **Dim** the **screen** of major **events** (e.g. low health)

Interface: Witcher 3-video-



Other Forms of Feedback

- **Sound**

- **Player** can determine type, distance
- In some **set-ups**, can determine direction
- Best for **conveying action** “off-screen”

- **Tactile** (e.g. **Rumble Shock**) **video**

- Good for proximity only (**near** vs. **far**)
- Either **on** or **off**; no type information
- Limit to **significant events** (e.g. **getting hit**)

Content and Challenges

- Content is **everything else**
- **Gameplay** content **defines the actual game**
 1. Goals and victory **conditions**
 2. Missions and quests
 3. Interactive **story** choices
- **Non-gameplay content affects player experience**
 1. Graphics and **cut scenes**
 2. Sound **effects** and background music
 3. Non-interactive **story**

Mechanics vs. Content

- **Content** is the **layout of a specific level**
 - Where the exit is **located**
 - The **number and types of enemies**
- **Mechanics describe what these do**
 - What **happens** when **player touches exit**
 - How the **enemies move** and hinder player



Mechanics vs. Content



Why the division?

- They are **not developed sequentially**
 - **Content** may **requires changes to game engine**
 - **Interface** is **changing until the very end**
- Intended to **organize** your **design**
 - **Engine**: **decisions** to be made early, **hard-code**
 - **Mechanics**: design **decisions**
 - **Interface**: how to shape the **user experience**
 - **Content**: specific **gameplay and level-design**

Summary

- Game is divided into four components
 - Should keep each in mind during design
 - Key for distributing work in your group
- But they are all interconnected
 - System/engine limits your possible mechanics
 - Content is limited by the type of mechanics
- Once again: **design is iterative**

Thank you





GAME programming

Lecture 4: unity 2D

Marwa Al-Hadi



Introduction to Unity 2D

Unity is available for both 2D and 3D games. When you create a new project in Unity, you will have a choice to start in 2D or 3D mode. The choice between starting from 2D or 3D mode determines some settings for the Unity Editor, such as whether images are imported as sprites or textures. You can swap between 2D or 3D mode at any time regardless of the mode you set when you created your project.

Sprites in Unity

Sprites are simple 2D graphic objects that have graphical images (called textures) on them. Unity handles sprites by default when the engine is in 2D mode.

If you are 3D, sprites are essentially just standard textures, but there are special techniques for combining and managing sprite textures for efficiency and convenience during development. When you view the sprite in 3D space, sprites will appear to be paper-thin, because they have no Z-width.

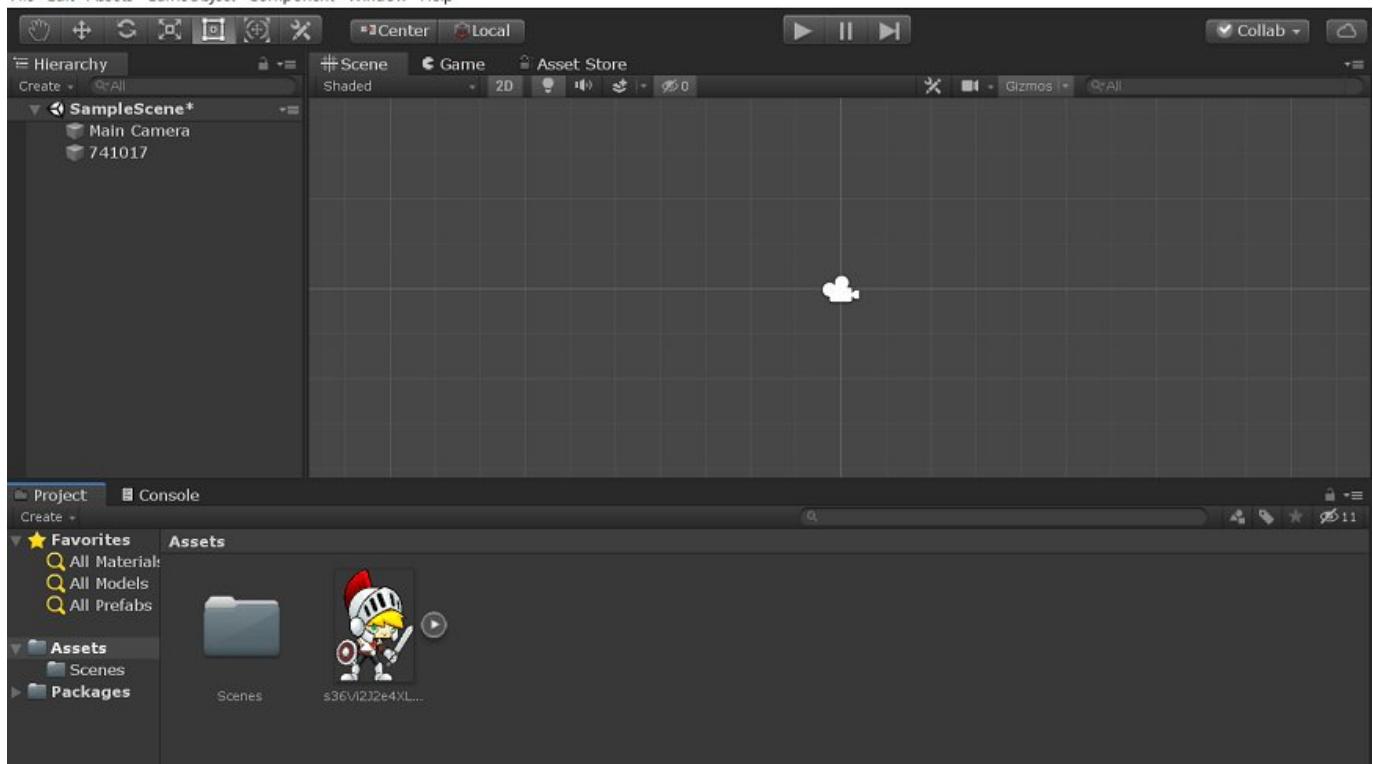
Sprites always face the camera at a right angle unless rotated in 3D space.

When you create a new sprite, it uses a texture. This texture is then applied on a fresh GameObject, and the Sprite Renderer component is attached to it. This makes our GameObject visible with our texture, as well as its properties related to how it looks on-screen.

Creating Sprites

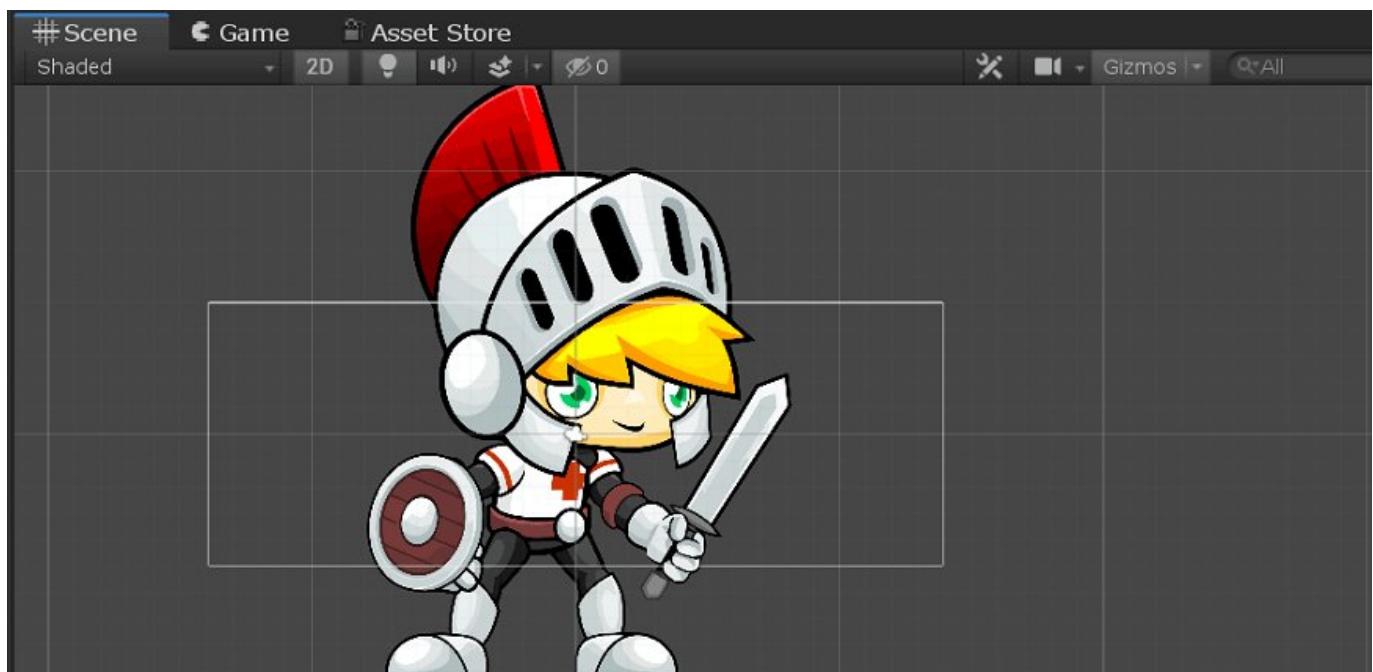
To create a sprite to your game, you must supply the engine with a texture. Let's create a texture first.

- Get an image what you want to add as a sprite in standard image file such as PNG or JPG that you want to use,
- Save it in your system directory and
- Then drag the image into the Assets region of Unity.



- Now drag the image from the Assets into the Scene Hierarchy.

You will notice that as soon as you let go of the mouse button, **a new GameObject with the name of the texture shows up in the list**. You will also get the image now in the middle of the scene in the scene view.



Let us consider the following points while adding a sprite:

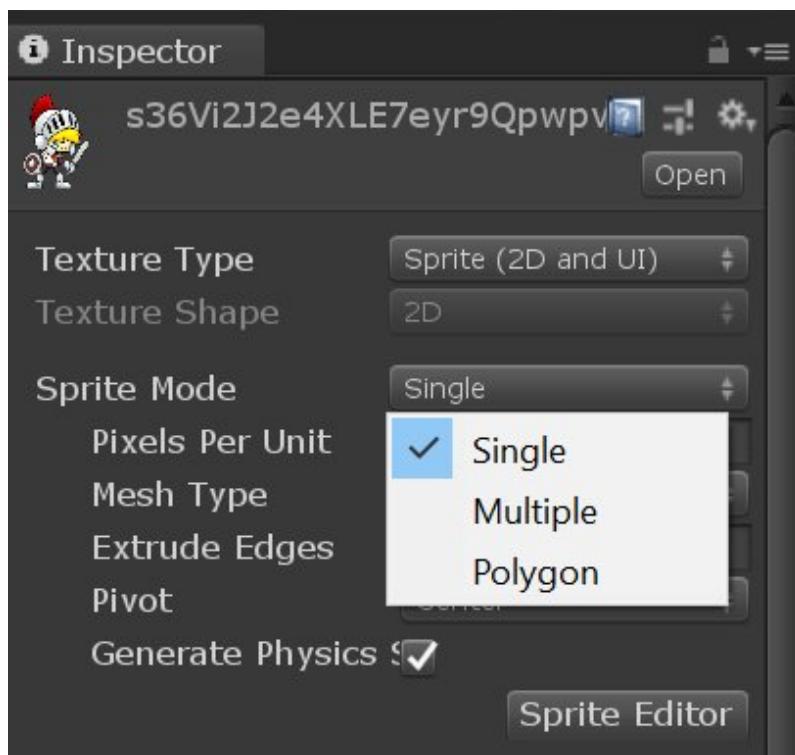
- By dragging from an external source into Unity, we are putting an asset.

- This added asset is an image, so it becomes a texture.
- By dragging this texture into the scene hierarchy, we are creating a new GameObject with the same name as our texture, with a sprite renderer attached.
- This sprite renderer uses that texture to draw the image in the game.

We have now added a sprite in our scene.

Sprite Modes

This setting is used to specify how the sprite graphic is extracted from the image. To choose the modes, click on a sprite in the Assets/ Sprites folder, in the inspector, there are three different modes in which you can use Sprites:



Single: It is used for a single image sprite.

Multiple: It is used for a sprite with multiple elements, such as animations or spritesheets, with different parts for a character.

Polygon: It is used for a custom polygon-shaped sprite that you can create many different types of primitive shapes with, for example, Square, Triangle, Pentagon, Hexagon, etc.

Modifying Sprites

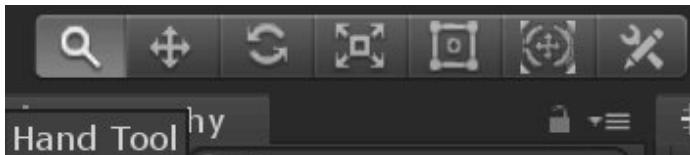
We can manipulate the imported sprites in various ways to change how it looks.

If you look at the top left corner of the unity interface, you will get a toolbar, as shown below:



Let's see the functions of these buttons:

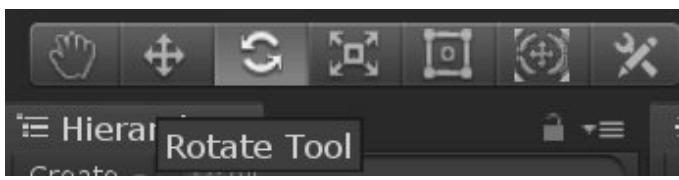
A **first-Hand** tool is used to move around the scene without affecting any objects.



The next tool is the **Move** tool. This is used to move the objects in the game world around.



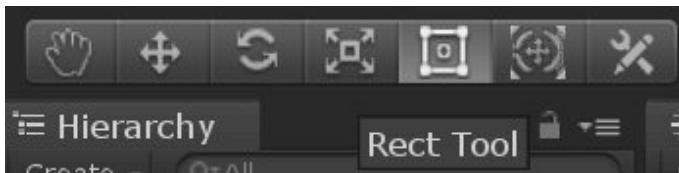
The next tool is the **Rotate** tool, which is used to rotate objects along the Z-axis of the game world or parent object.



The centered tool is the **Scale** tool. This tool allows you to modify the size (scale) of the objects along certain axes.

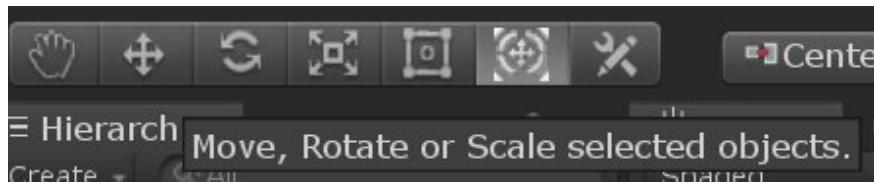


The next tool is the **Rect** tool. This tool behaves like a combination of the Move and the Scaling tool but is prone to loss of accuracy. It is more useful in arranging the UI elements.

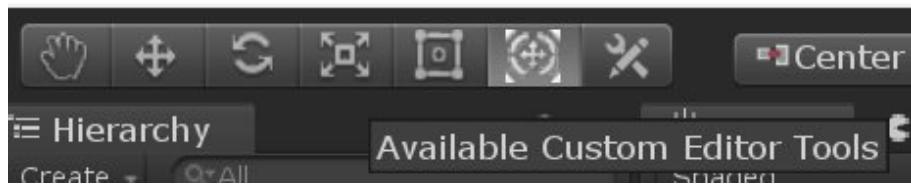


The next tool is the **Move, Rotate, and a Scale** tool. It is used to move, rotate, and scale the

selected object.



And finally, the last tool is the **Custom Editor** tool.



These tools are very useful and worthy as the complexity of the project increases.

← Prev

Next →

[Youtube](#) For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

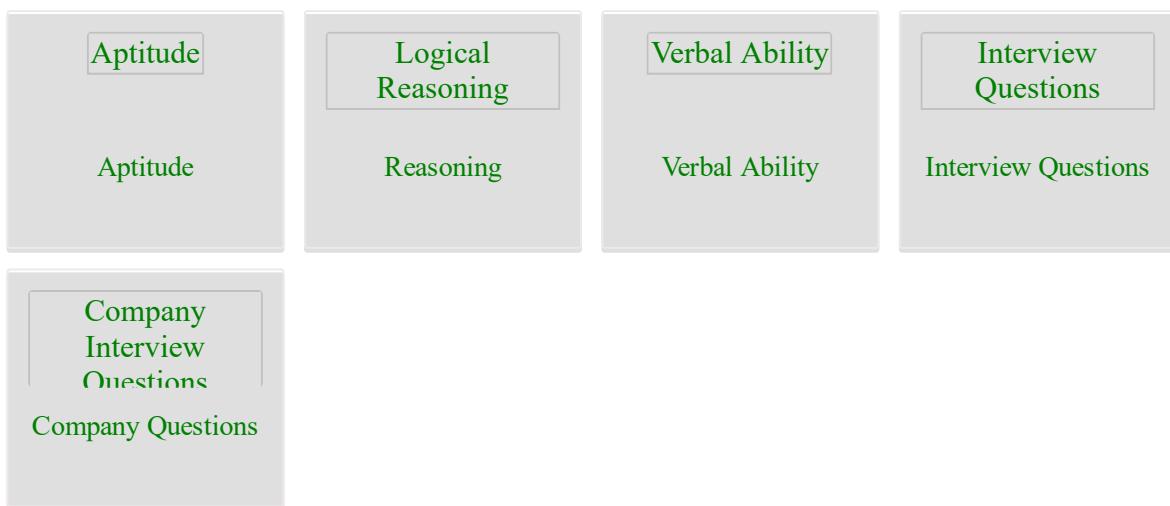
Swagger

[T-SQL tutorial](#)

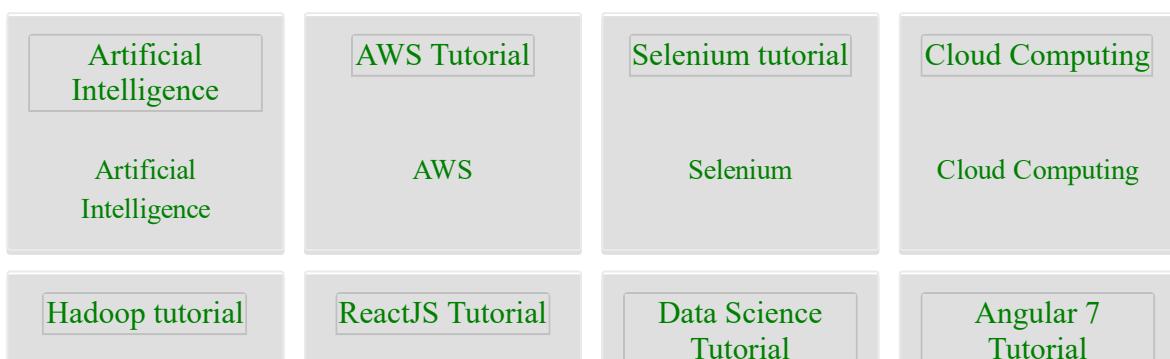
Transact-SQL



Preparation



Trending Technologies



Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial
Cyber Security	Automata	C Programming	C++
Java tutorial	.Net Framework tutorial	Python tutorial	List of Programs
Java	.Net	Python	Programs
Control Systems tutorial	Data Mining Tutorial	Data Warehouse Tutorial	

Control System

Data Mining

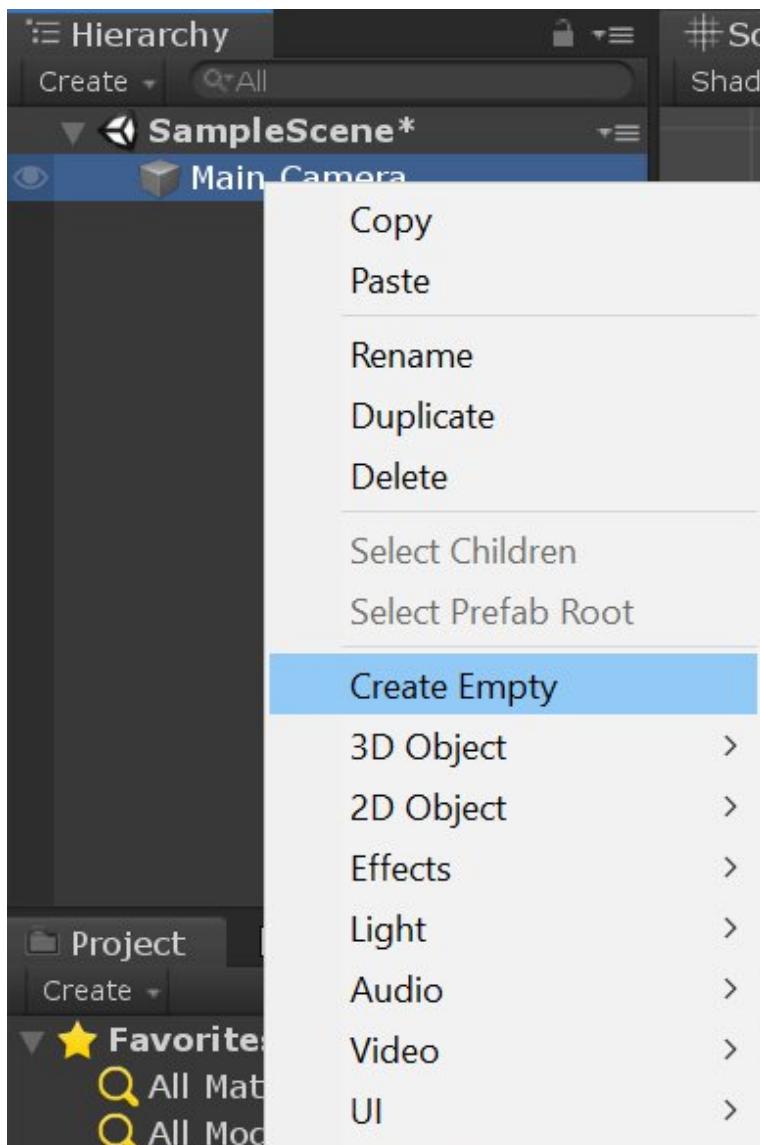
Data Warehouse

2D Sprite Sheet

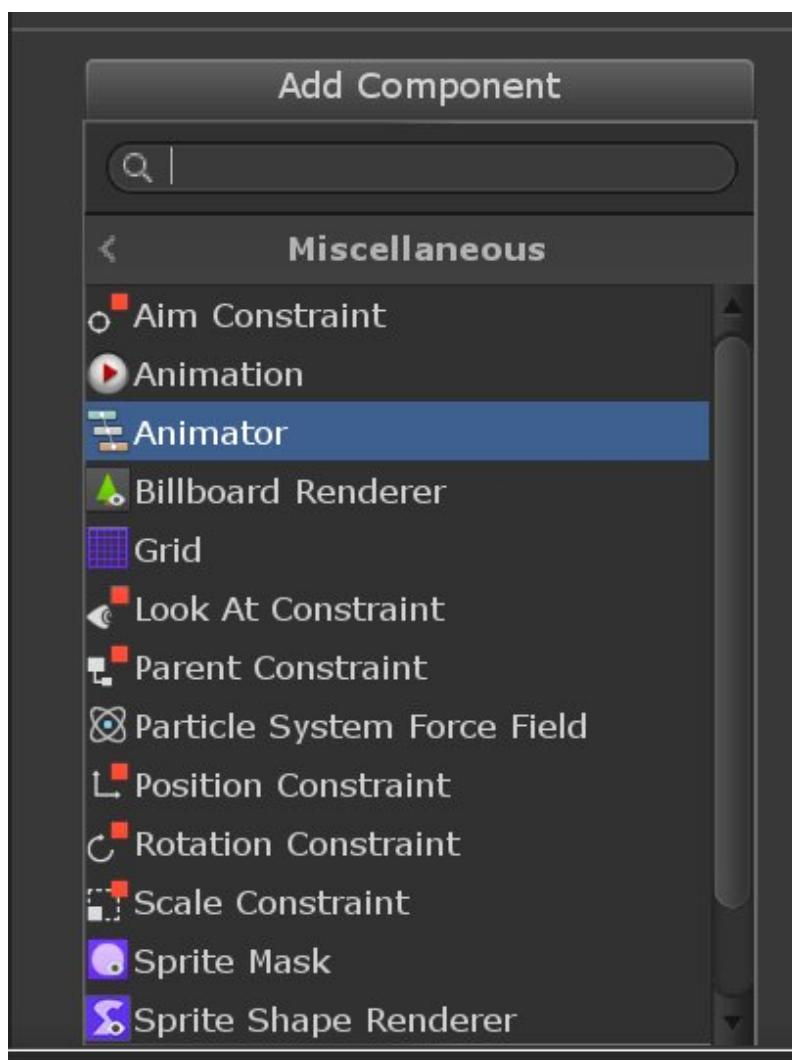
In a nutshell, a **sprite sheet** is a **way of packing images together as one image**, which is then used **to create animations and sprite graphics** as it will use **low memory and increase the performance** of games.

Creating GameObject and Adding Components

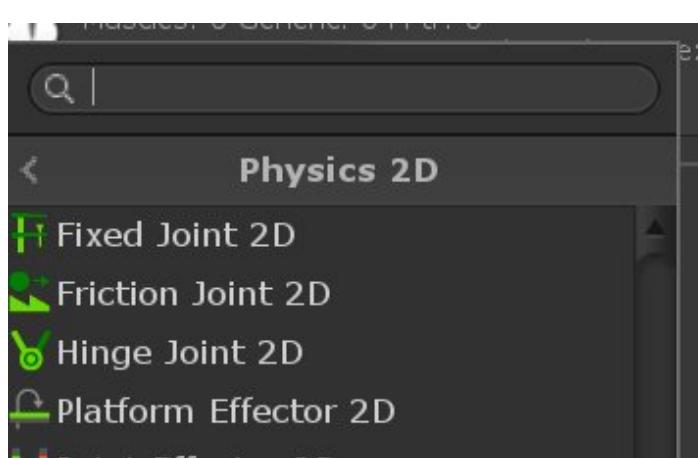
- First of all, we need a GameObject in our scene. For this Right Click on the Hierarchy tab and select Create Empty.

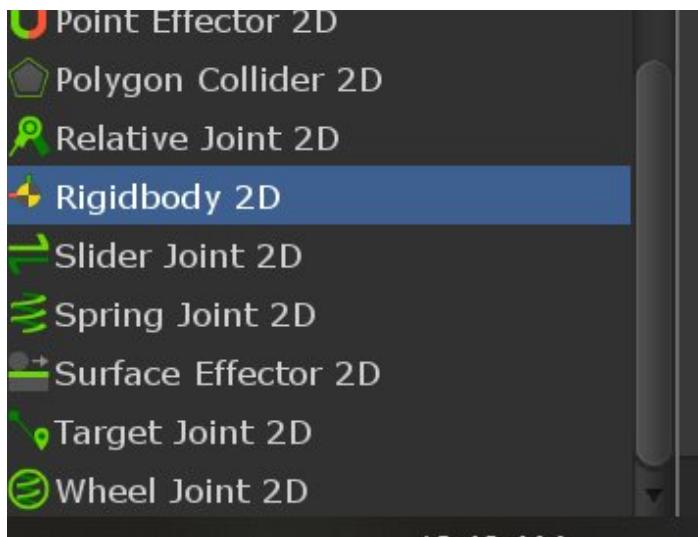


- o Rename the new GameObject. Here, we renamed it to Player.
- o Select the Player GameObject and Go to the Inspector tab. In the Inspector tab, click the Add Component button. Choose miscellaneous -> Animator. Make sure to choose Animator nor Animation.

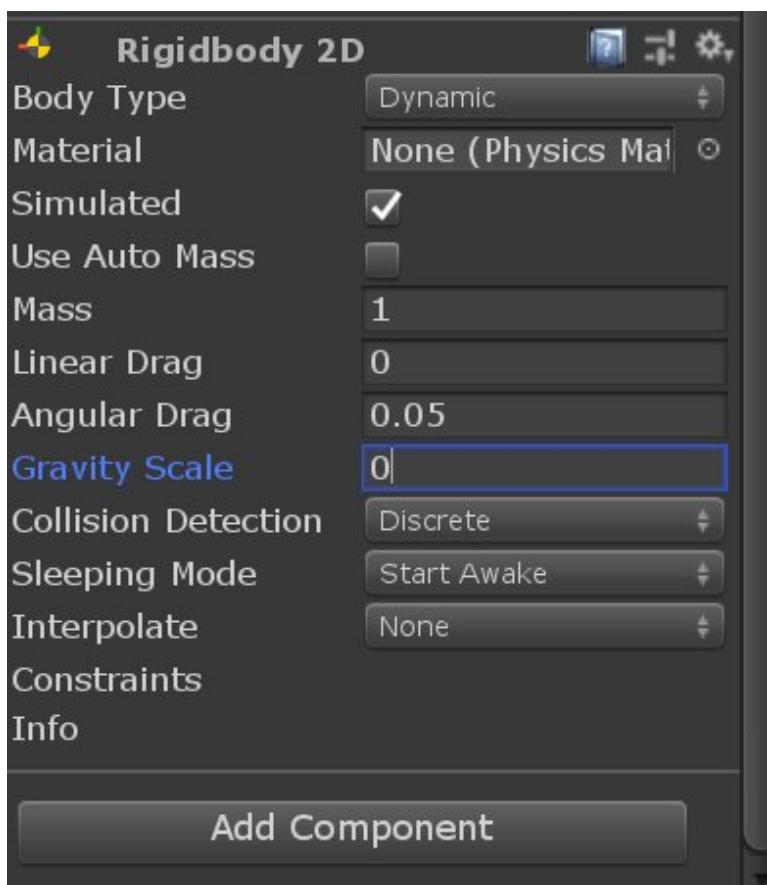


- o Add another component, **Rigidbody2D**. For this, click on the Add Component button and choose Physics 2D -> Rigidbody 2D.





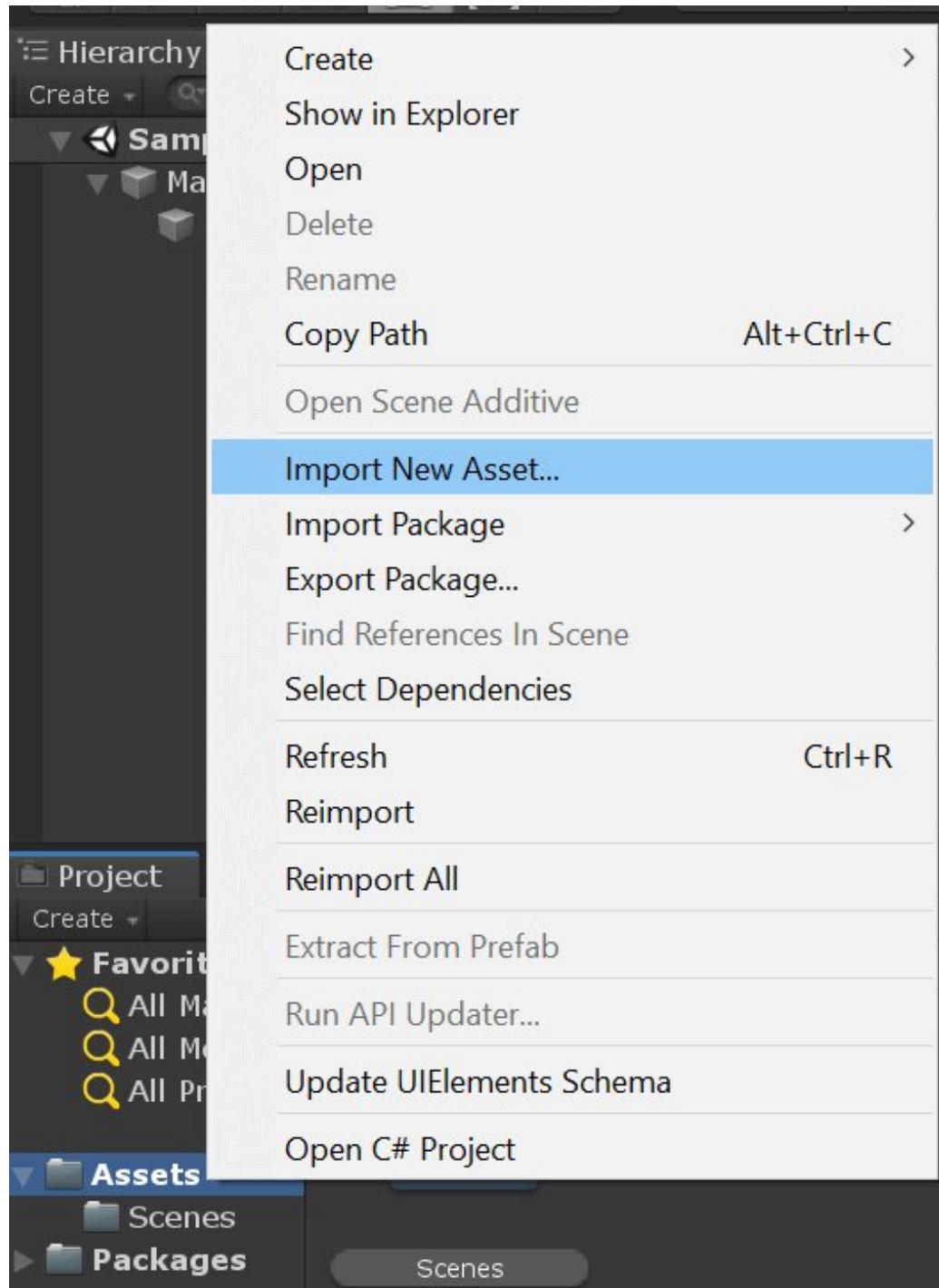
- Now, in the Rigidbody 2D component, set the Gravity Scale to zero.



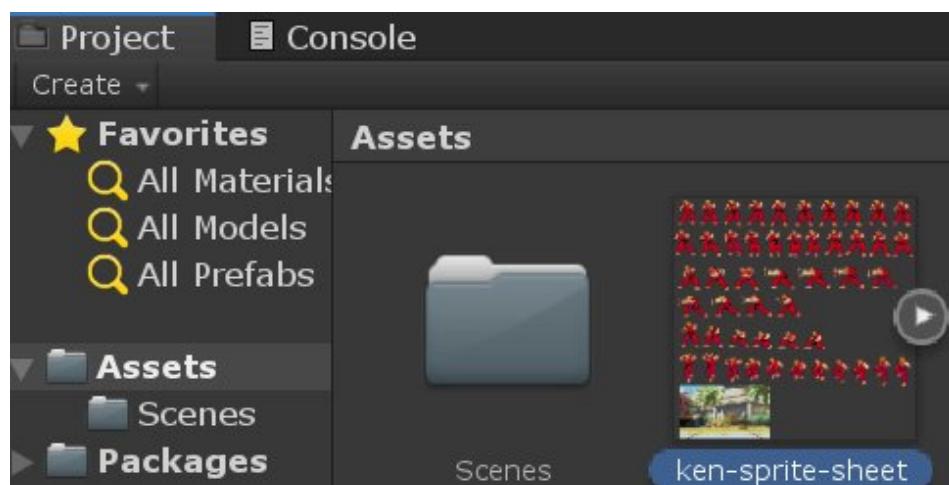
Importing the Sprite Sheet

- First of all, download the sprite sheet that contains all the frames of animation for a simple walking animation. A link is given below from where you can download: [ken-sprite-sheet.png](#)
- Right-click on the Assets and select Import New Asset...

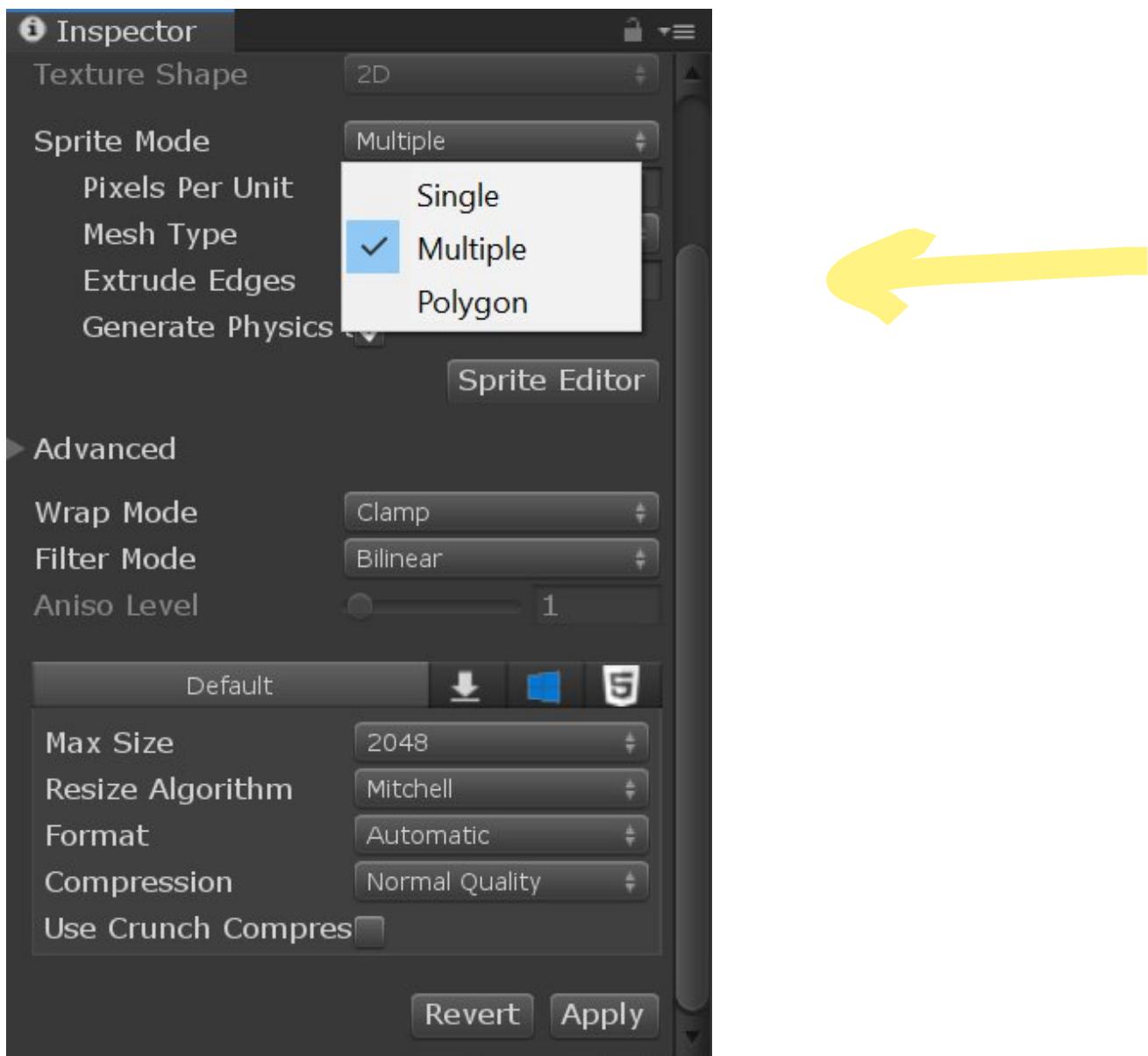




- Browse to the sprite sheet image you downloaded and click the import button.



- Select this imported Asset, and in the Inspector window, **change the Sprite mode option from Single to Multiple**. Click the Apply button in the Inspector tab.



Now Unity will treat ken-sprite-sheet.png as a sprite sheet with multiple frames of animation.

Slicing Sprite Sheet

- Go to the Windows menu and select 2D -> Sprite Editor.

↳ Linux Standalone - Unity 2019.2.6f1* <DX11>

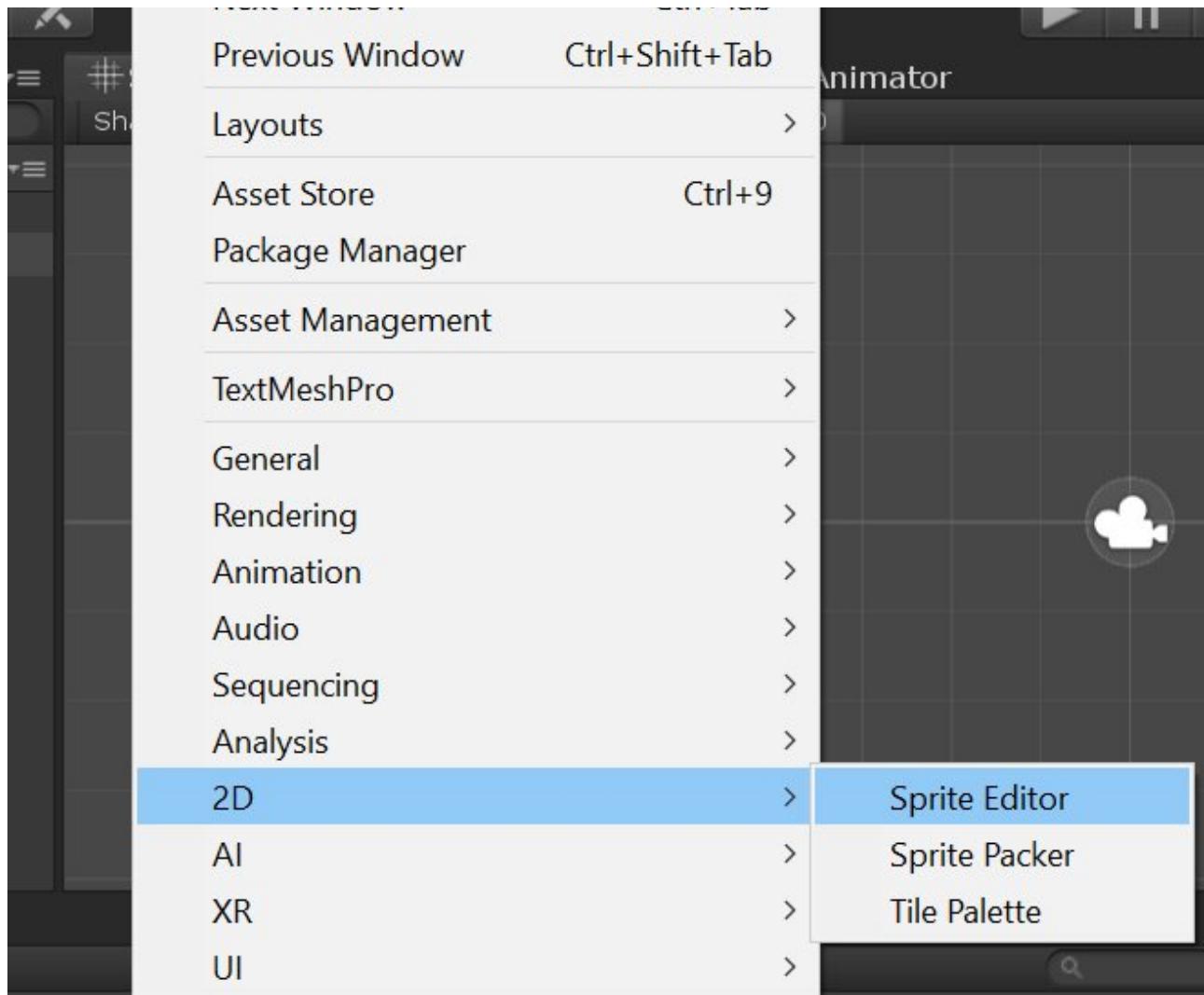
Component Window Help



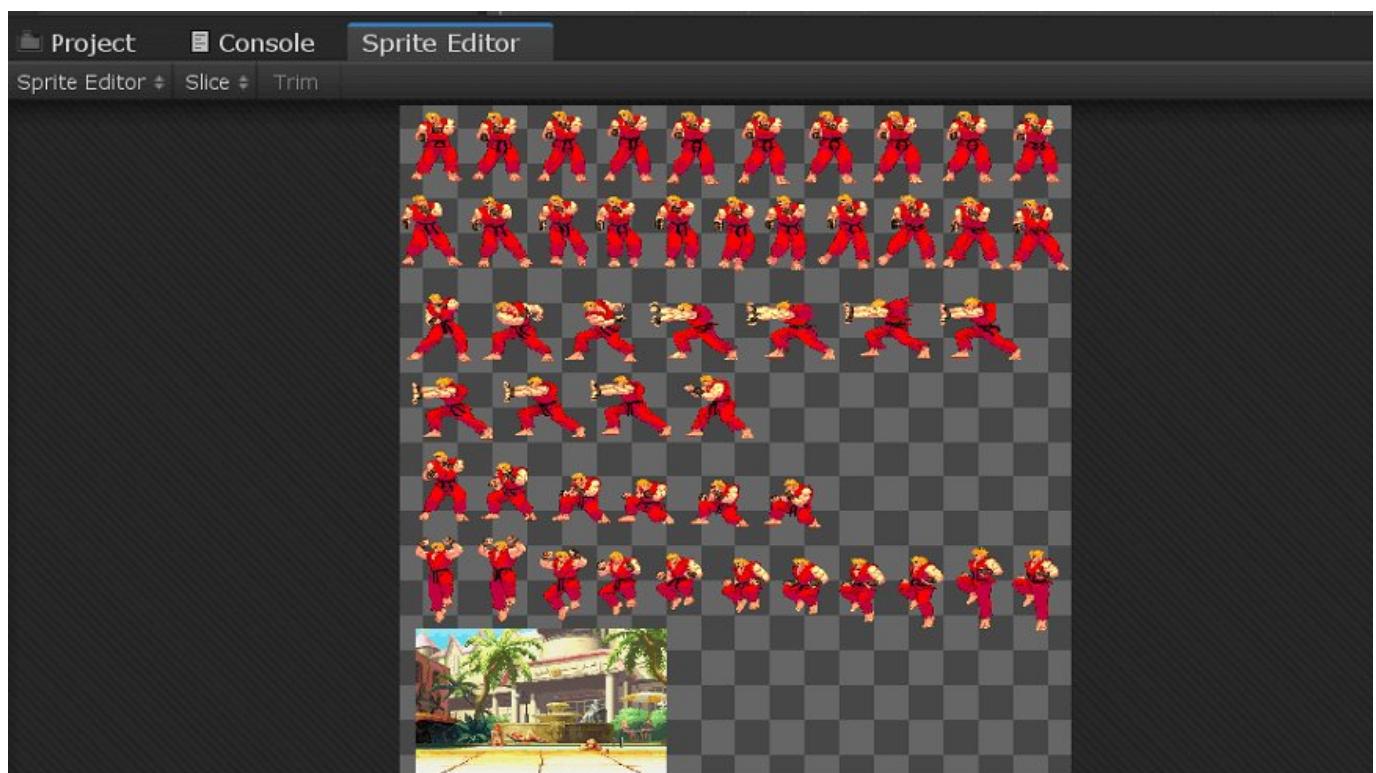
Next Window

Ctrl+Tab



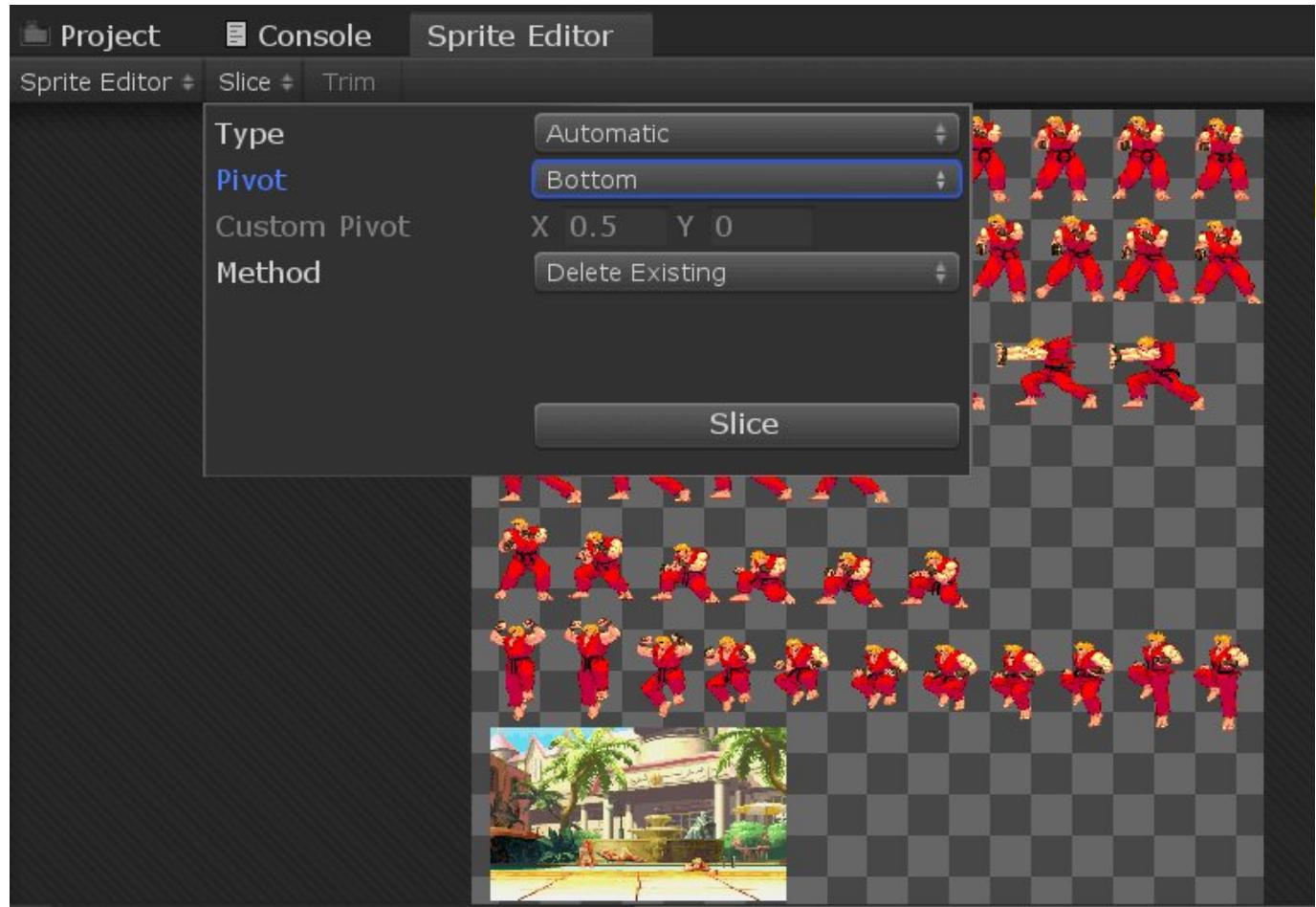


- o Drag the **Sprite Editor** window and dock it. I docked mine alongside the **Console** tab.



- Click on the Slice drop-down, and here we can see that the options Automatic, Centre, and Delete Existing are default chosen. Some sprite-sheets have their images ordered in different ways. That is why there are a number of options to choose from.

Here I changed the Pivot value from Centre to Bottom. When you change the Pivot to Bottom means it sets the pivot point to the center bottom of the sprite, and slicing Type in this particular case should be set to be automatic.



- Then click on the Slice button. Unity has now separated all the sprites. Each sprite should have its own bounding box, clicking them provide information on each sprite and allow fine-tuning of sizes and pivot points.

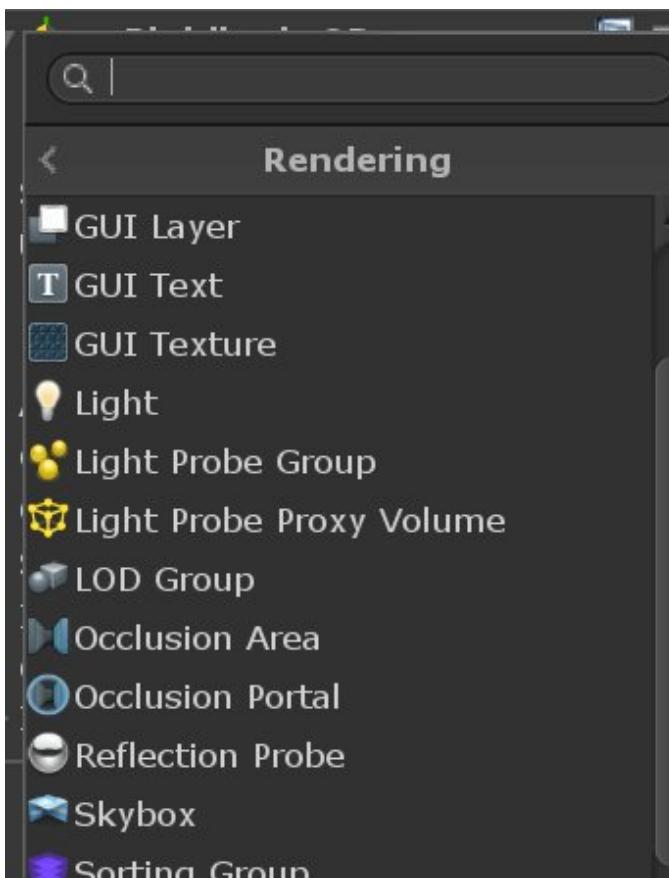


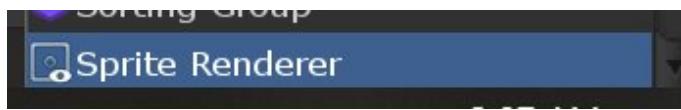


- Now, click on **Apply** in the top right corner of the Sprite Editor window.

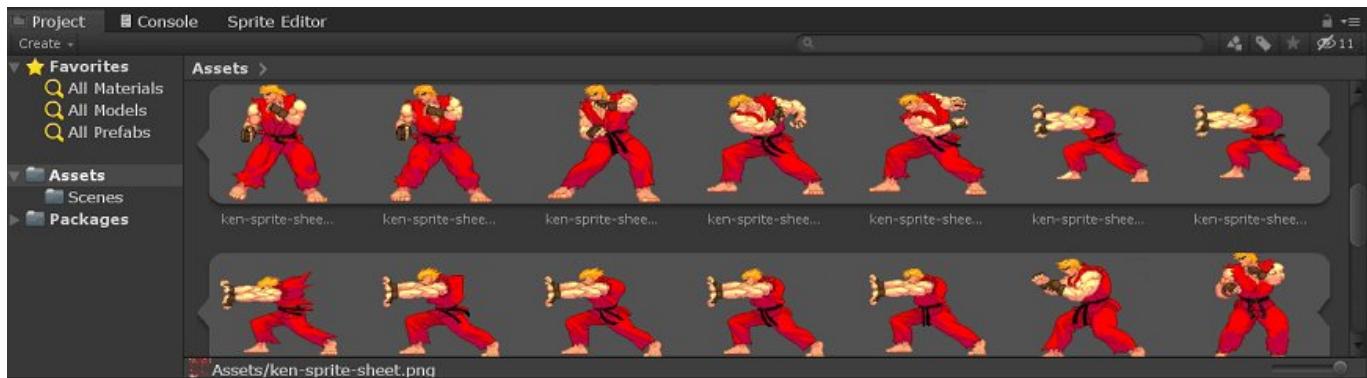


- Select the Player object in the Hierarchy tab and then in the Inspector tab click on **Add Component** button. Then select **Rendering -> Sprite Renderer**.





- Finally, you will be able to see all of your Sprites as an individual object in the Project -> Assets Folder.



← Prev

Next →

[Youtube](#) For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

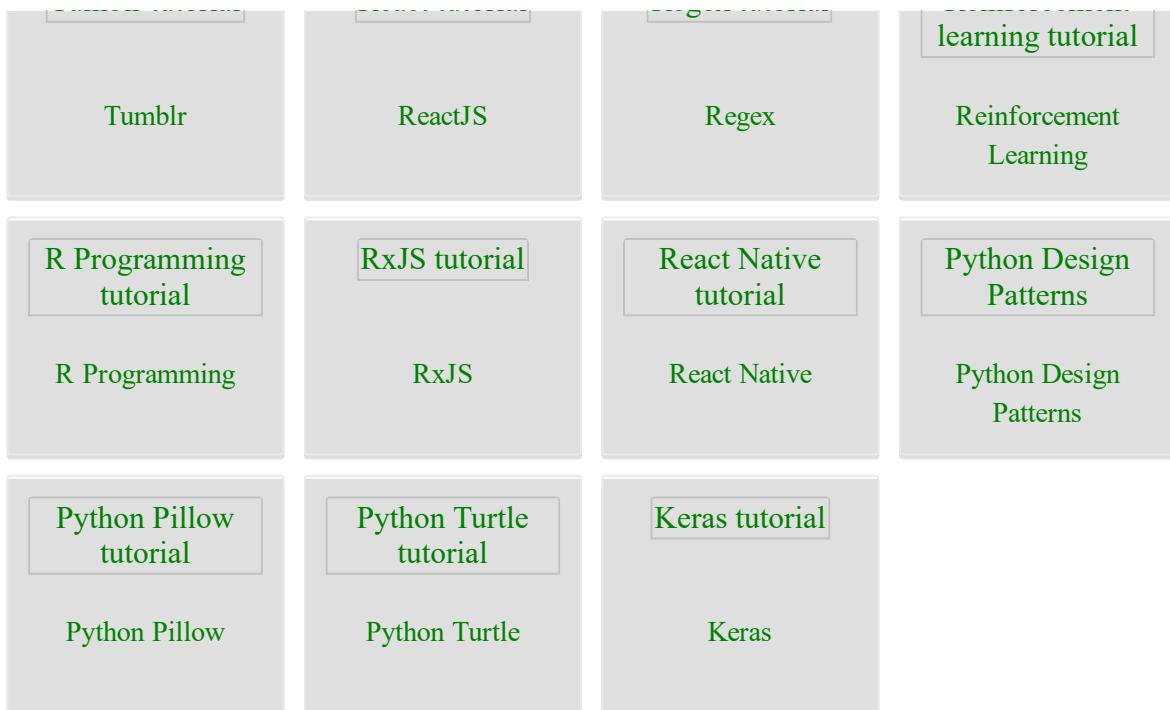
Transact-SQL

[Tumblr tutorial](#)

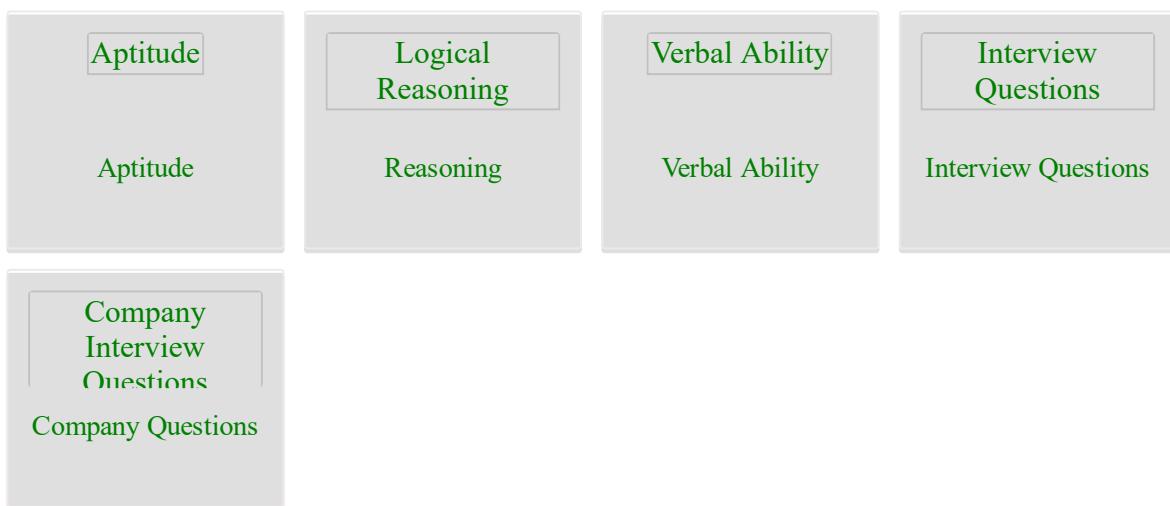
[React tutorial](#)

[Regex tutorial](#)

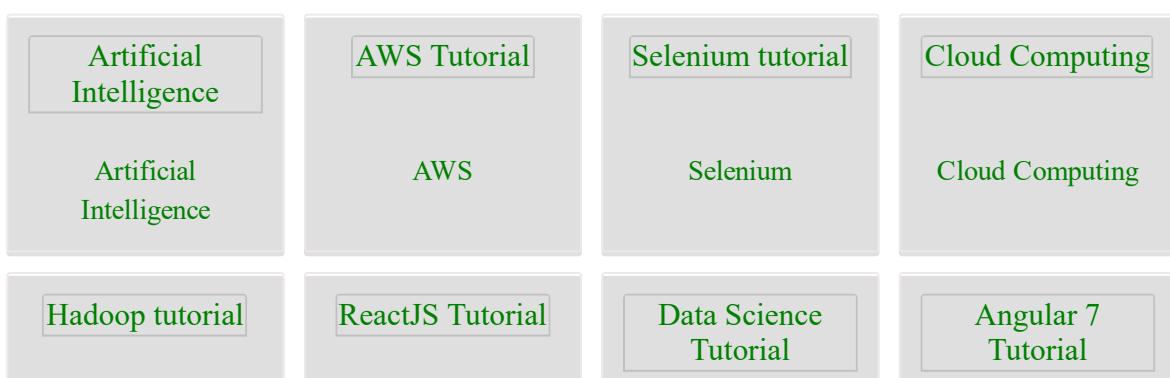
[Reinforcement](#)



Preparation



Trending Technologies



Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial
Cyber Security	Automata	C Programming	C++
Java tutorial	.Net Framework tutorial	Python tutorial	List of Programs
Java	.Net	Python	Programs
Control Systems tutorial	Data Mining Tutorial	Data Warehouse Tutorial	

Control System

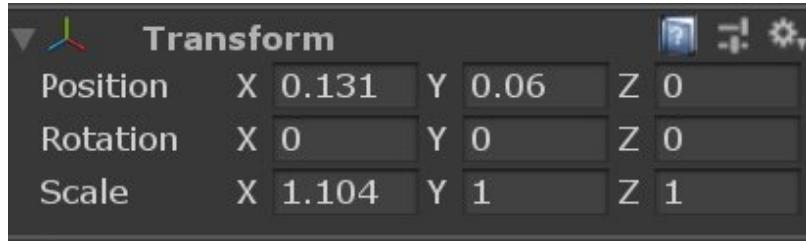
Data Mining

Data Warehouse

Transforms and Object Parenting

In Unity, the **Transform** component has three visible properties - the **position, rotation, and scale**.

Each of these properties has three values for the three axes. Means, Transform is used to determine the Position, Rotation, and Scale of each object in the scene. Every GameObject has a Transform.



Properties

Position: This is the **position of the transform in X, Y, and Z coordinates**. 2D games generally do not focus on the Z-axis when it comes to positioning. The most frequent use of the Z-axis in 2D games is in the creation of parallax.

Rotation: This property defines the amount of rotation (measured in degree) an object is rotated about that axis **with respect to the game world or the parent object**.

Scale: The scale of the object defines **how large it is when compared to its original or native size**. For example, let us take a **square of 2x2 dimensions**. If the square is scaled against the X-axis by 3 and the Y-axis by 2 we **will get a square of size 6x4**.

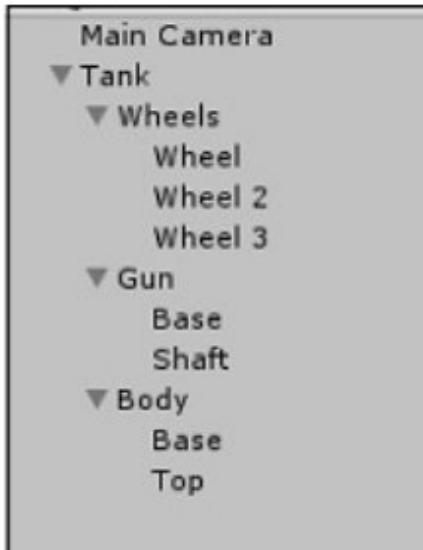
These properties are measured relative to the transform's parent. If the transform has no parent, the properties are calculated in world space.

Object Parenting

In Unity, **GameObjects** follow a **Hierarchy system**. Using this hierarchy system, **GameObjects can become parents of other GameObjects**. When a GameObject has a parent, it will perform all its transform changes with respect to another GameObject instead of the game world.

A parent object causes all children objects to move and rotate the same way the parent object does, although moving children objects does not have any effect on the parent. Children themselves can be parents; e.g., your hand is the child of your arm, and fingers are children of your hand.

Parenting GameObjects has a number of uses. For example, all the different parts of a tank could be separate GameObjects, parented under a single GameObject named "tank." Hence, when this "tank" parent GameObject moves, all the parts move along with it because their positioning is updated constantly according to their parent.



← Prev

Next →

Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

Aptitude

[Logical Reasoning](#)

Reasoning

[Verbal Ability](#)

Verbal Ability

[Interview Questions](#)

Interview Questions

[Company Interview Questions](#)

Company Questions

Trending Technologies

Artificial Intelligence	AWS Tutorial	Selenium tutorial	Cloud Computing
Artificial Intelligence	AWS	Selenium	Cloud Computing
Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular 7 Tutorial
Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial
Cyber Security	Automata	C Programming	C++

Java tutorial

Java

.Net Framework
tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

Data Warehouse
Tutorial

Data Warehouse

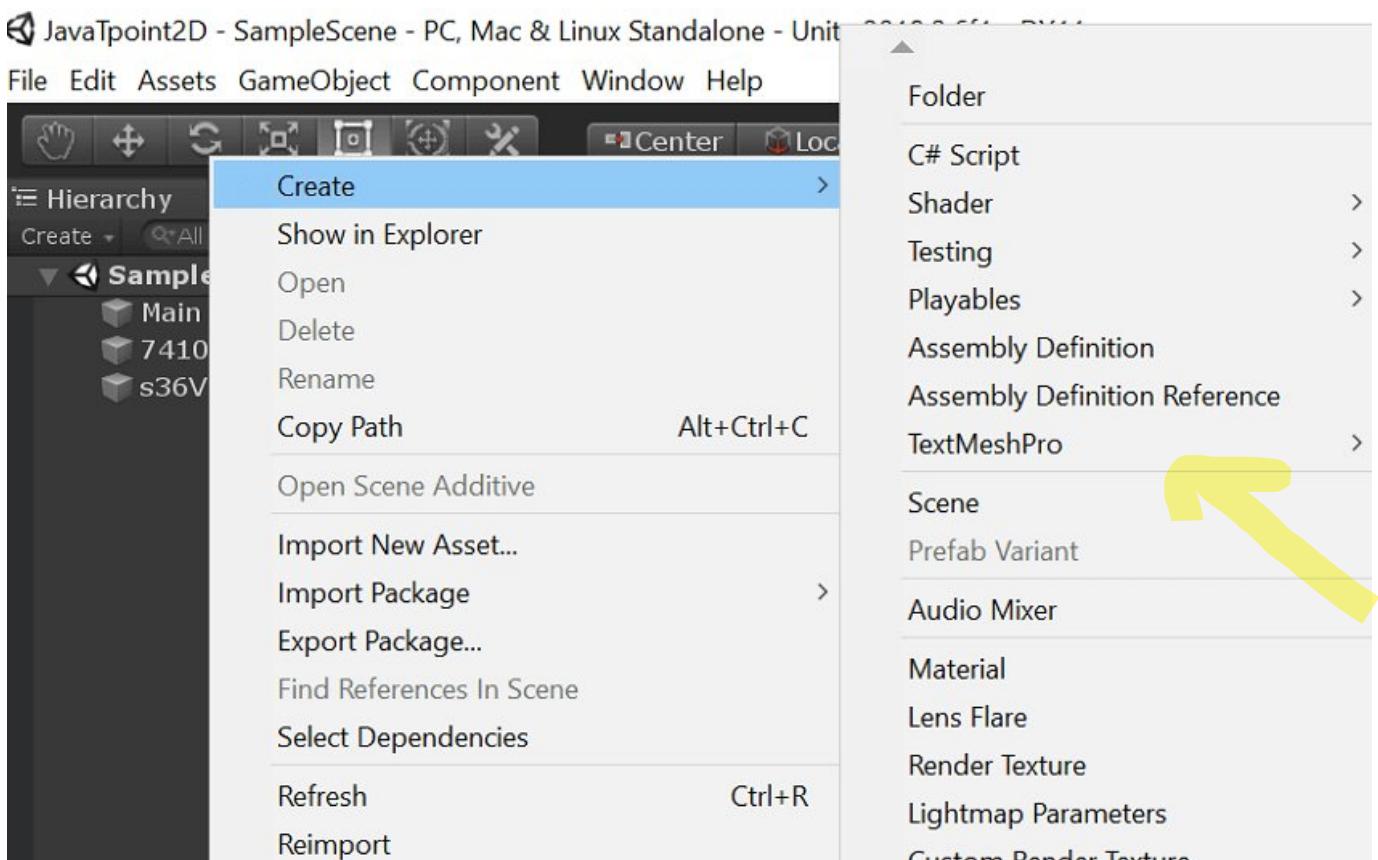
Internal Assets

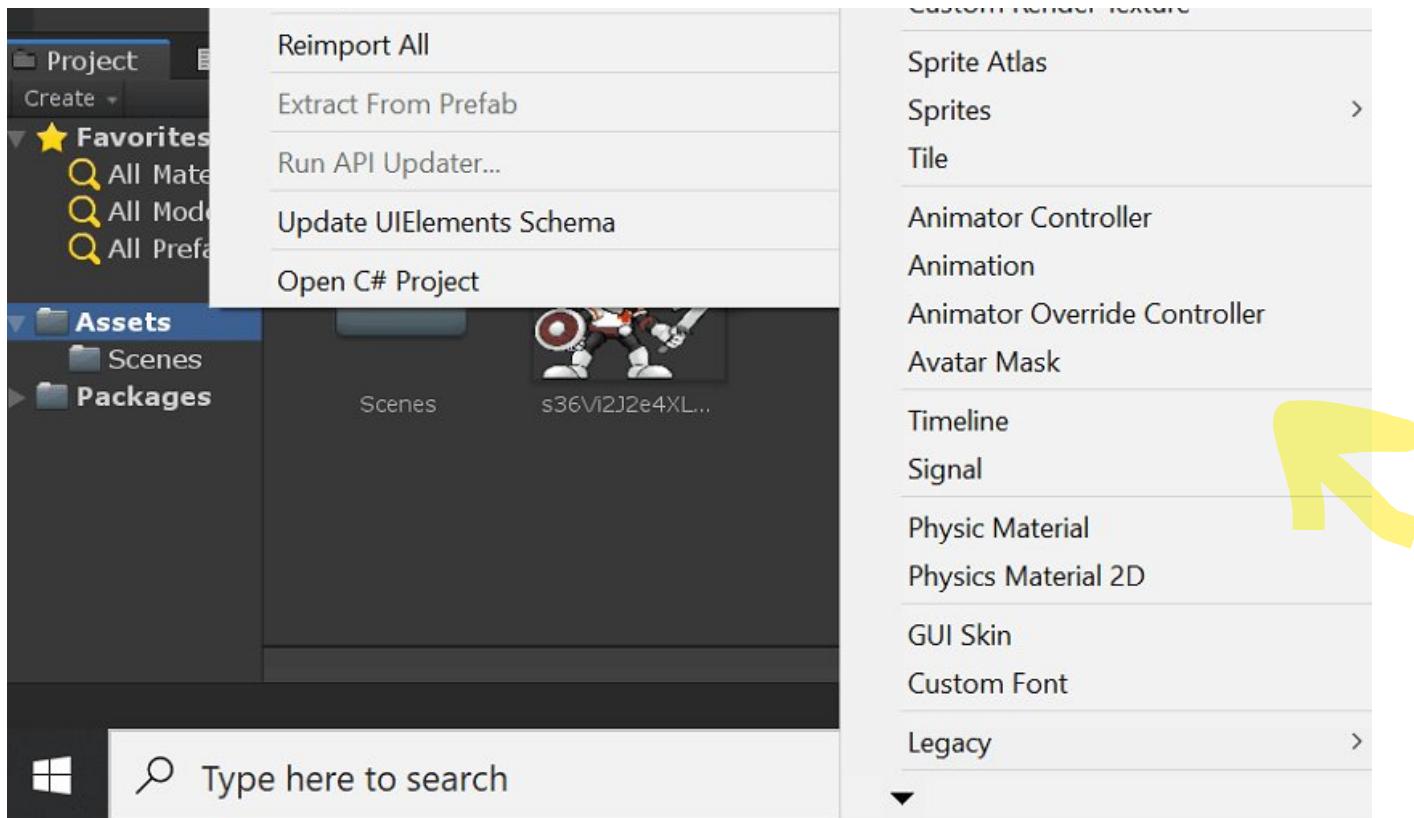
Along with the **external assets** that you import from other programs such as image, audio files, 3D models, etc., Unity also offers the creation of internal assets. These assets are formed within Unity itself, and as such, do not need any external program to create or modify.

Let's see some examples of **internal assets**:

- **Scenes:** These act as levels.
- **Animations:** These contain data for the animation of GameObject.
- **Materials:** These are used to define how lighting affects the appearance of an object.
- **Scripts:** The code which will be written for the GameObjects.
- **Prefabs:** These act as a blueprint for GameObjects so they can be generated at runtime.

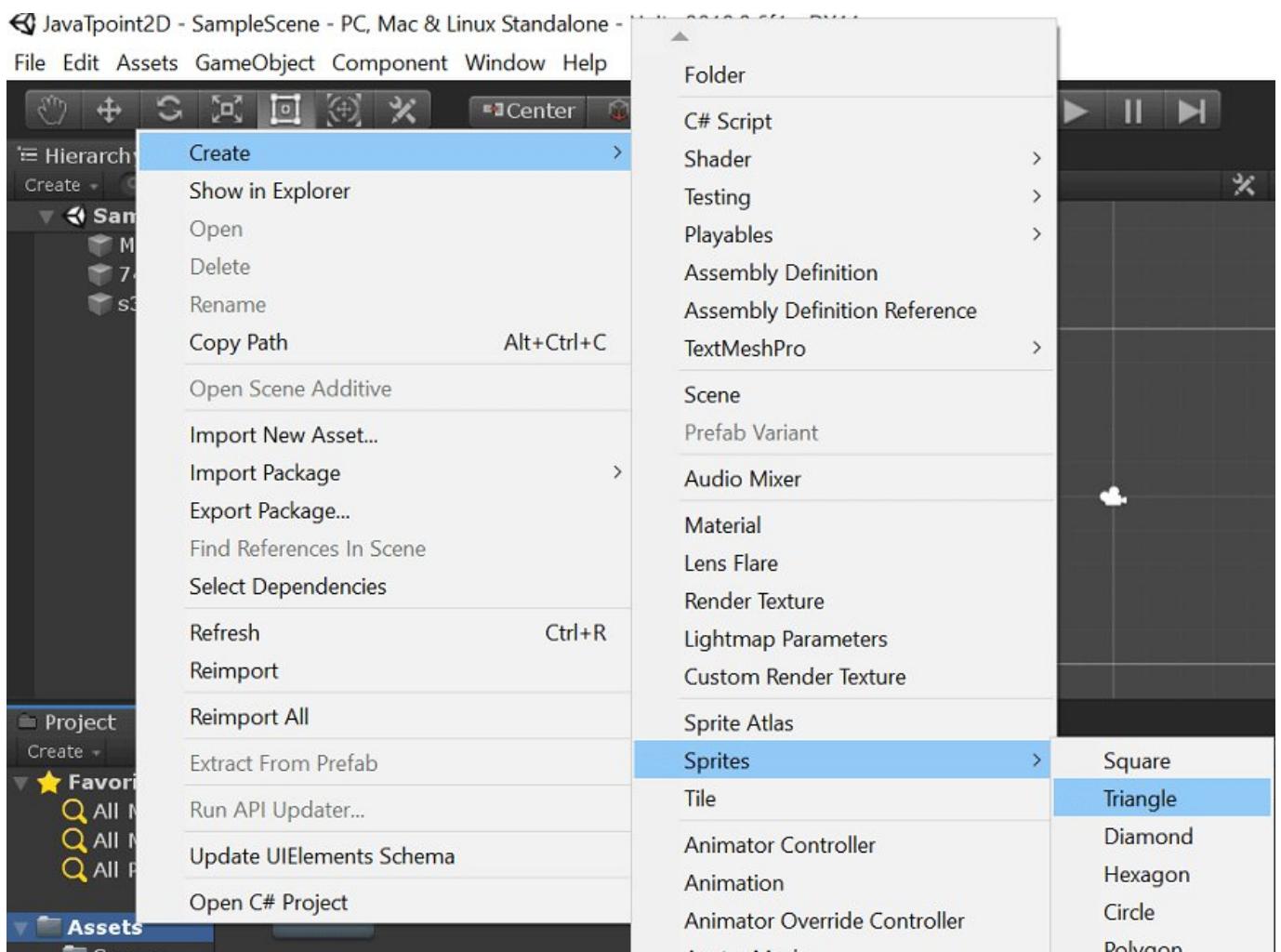
To create an internal asset, go to the Assets folder and right-click on that folder and select **Create**.

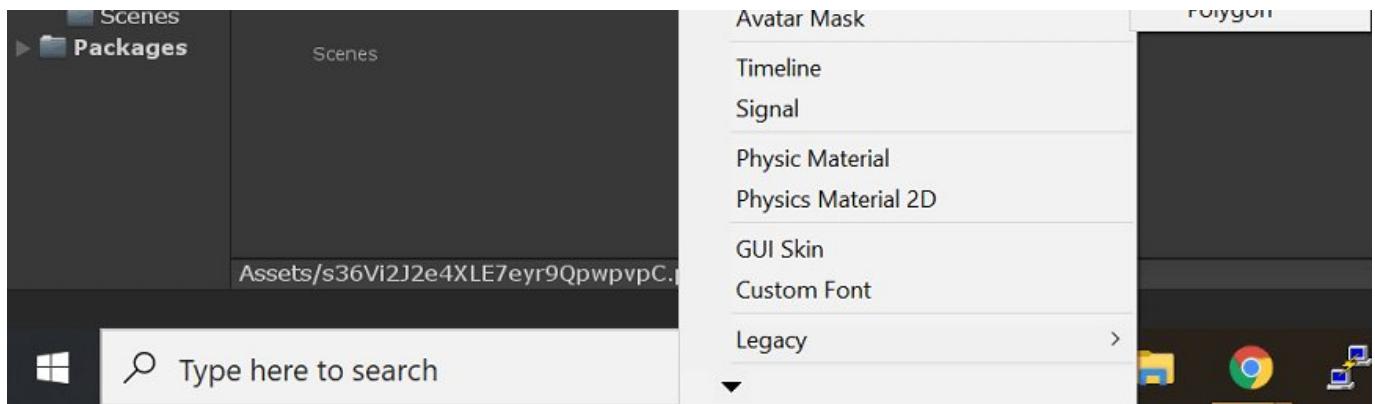




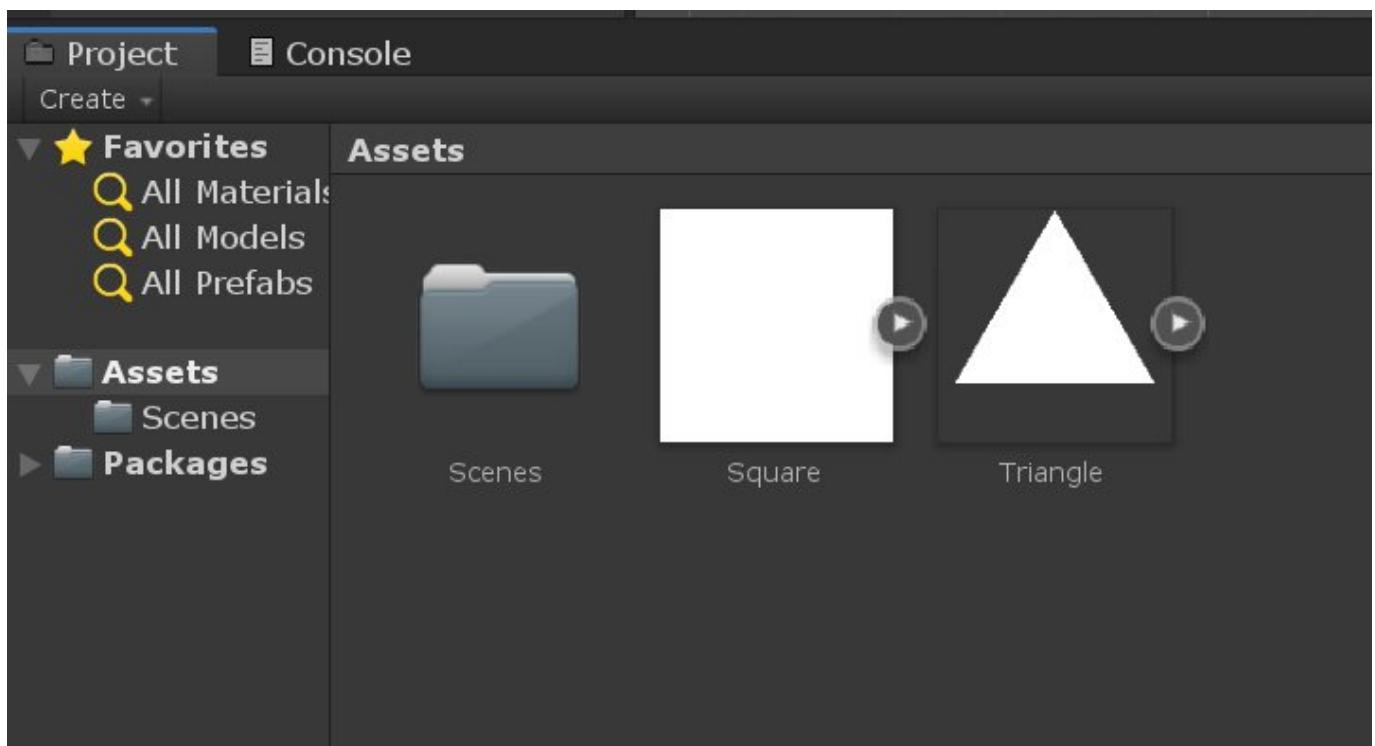
In this example, we will create a **Triangle** and a **Square**.

To create a **Triangle** asset, right click on the asset -> **Create** -> **Sprites** -> **Triangle**





Repeat the same process for Square, and now you should have two new graphic assets.



As we move along, we will explore more of these internal assets, since they are very crucial to build a proper game.

← Prev

Next →

Youtube For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

Aptitude

[Logical Reasoning](#)

Reasoning

[Verbal Ability](#)

Verbal Ability

[Interview Questions](#)

Interview Questions

[Company Interview](#)

Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Computer Organization

Discrete Mathematics Tutorial

Discrete Mathematics

Ethical Hacking

Computer Graphics Tutorial

Software Engineering

html tutorial

Ethical Hacking

Computer Graphics

Software
Engineering

Web Technology

Cyber Security
tutorial

Cyber Security

Automata
Tutorial

Automata

C Language
tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework
tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

Data Warehouse
Tutorial

Data Warehouse



Togoppi

Game programming

Lecture 5,6: Unity scripting C#

Presented by

Marwa Al-Hadi



Ash

Variables and Functions

GameObjects have components that make them behave in a certain way. When we refer to components, we are referring to the available functions of a GameObject, for example, the human body has many functions, such as moving, eating, talking, and observing. Now let's say that we want the human body to move faster. That means movement function is linked to that action. So to make our human body to move faster, we would need to create a script that had access to the movement component, and then we would use that to make the body move faster.

Just like in real life, different GameObjects can also have different components; for example, the camera component can only be accessed from a camera. Many components already exist that were created by Unity's programmers, but we can also create our own components.

This means that all the properties that we see in the Inspector tab are just variables of some type. They simply store data that will be used by some function.

Public Variables

If you put the public at the beginning of a variable statement means that the variable will be visible and accessible. It will be visible as a property in the Inspector panel so that you can change the value stored in the variable.

Example:

```
public int num = 10;
```

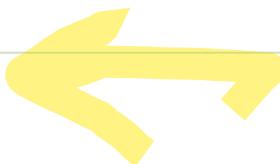


Private Variables

Not all the variables needed to be public. If there is no need for a variable to be changed in the Inspector panel or be accessed from other scripts.

Example:

```
private int myNum = 2;
```



Example

Let's create a script with the name `myTest.cs` that has one int variable:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
    mono Behavior in Unity
    class provides the framework.
    1. allows you to attach your script to a Game Object in the editor,
    2. providing a connection into useful Events such as Start and Update

public class myTest : MonoBehaviour
{
    int num = 24;
    mono Behavior in Unity
    class provides the framework.

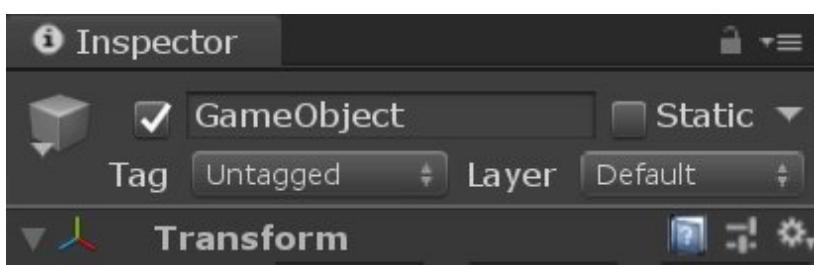
    void Start()
    {

    }

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



Now create a `GameObject` (from the top menu `GameObject-> Create Empty`) and add the `myTest.cs` script file to the `GameObject` (in the Inspector panel of newly created `GameObject`). Now it will look in the `GameObject`'s Inspector:





Here, we can see that it has `myTest.cs` script file, but it does not have our `num` variable at all.

Let's modify our script by adding a new variable, but this time with the `public` prefix. In a programming language, the `public` means that the value can be `seen by other classes too`. In Unity, the `public` also means that the variable will be shown in the Inspector.

The modified `myTest.cs` script is:

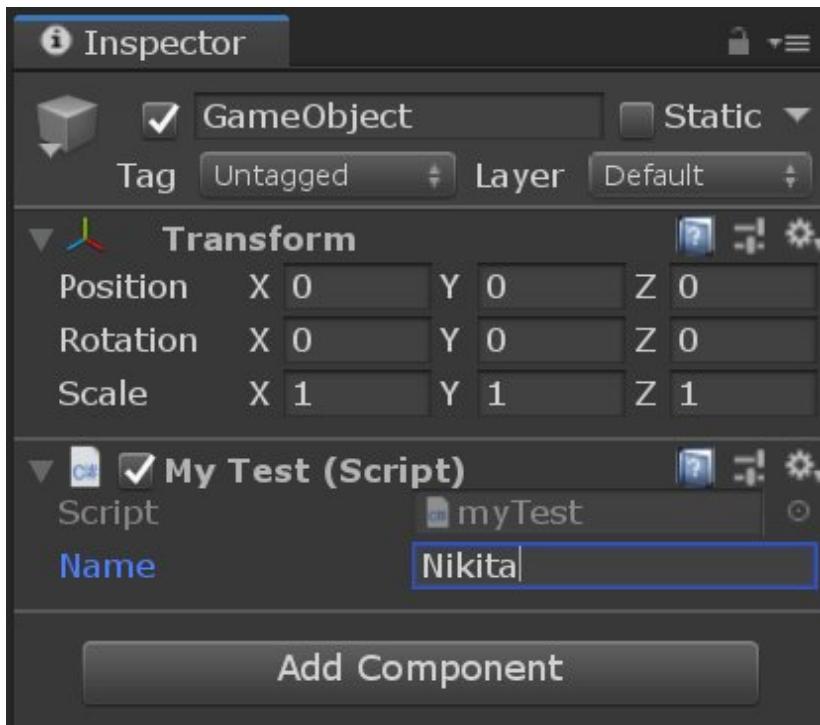
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class myTest : MonoBehaviour
{
    int num = 24;
    public string name;
    void Start()
    {
        Debug.Log("My name is " + name);
    }

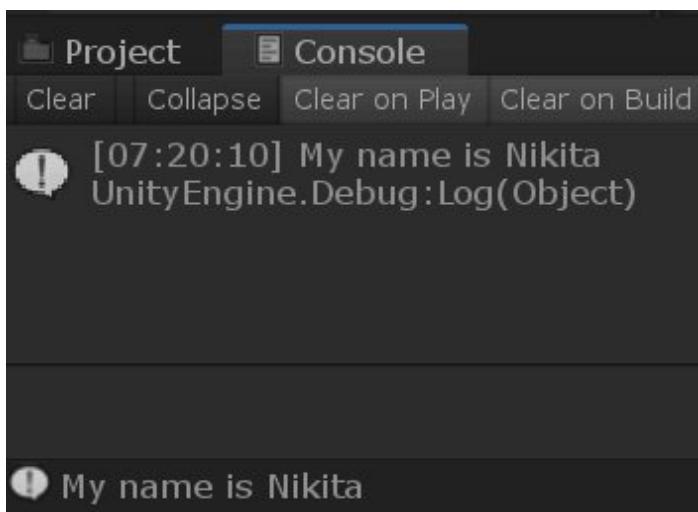
    // Update is called once per frame
    void Update()
    {
    }
}
```



Just like other components often have properties that are editable in the Inspector panel, we can allow values in our script to be edited from the Inspector too.



Enter the value for variable Name and then press play. You will see that the message includes the text you entered.



Assignment

1. create a script to write the scenario of game that appear at the beginning of the game

Hence, in Unity, you must declare a variable as public to see it in the Inspector.

← Prev

Next →

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

Splunk tutorial Splunk	SPSS tutorial SPSS	Swagger tutorial Swagger	T-SQL tutorial Transact-SQL
Tumblr tutorial Tumblr	React tutorial ReactJS	Regex tutorial Regex	Reinforcement learning tutorial Reinforcement Learning
R Programming tutorial R Programming	RxJS tutorial RxJS	React Native tutorial React Native	Python Design Patterns Python Design Patterns
Python Pillow tutorial Python Pillow	Python Turtle tutorial Python Turtle	Keras tutorial Keras	

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer

Compiler Design

Computer

Discrete

Network tutorial

Computer Network

tutorial

Compiler Design

Organization and Architecture

Computer Organization

Mathematics Tutorial

Discrete Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse

Conventions and Syntax

=rules for writing code

There are **some rules used in writing a good working code**. These rules are **called conventions**. Like, you need **capital letters**, **full stops**, and **commas** for a written sentence **to make sense**. The code has its syntax for the computer to understand what your code means.

See this block of code:

```
public class BasicSyntax : MonoBehaviour
{
    void Start ()
    {
        Debug.Log(transform.position.x);
    }
}
```

Curly Brackets {}

Curly brackets are used to **contain blocks of code inside specific functions**. Programming statements like if statements include curly brackets, as they can contain further blocks of code. As such, functions will often have code nested within code, separated by sets of brackets {}.

Dot Operator .

In the above part of the code, the **dot operator** is placed in between the words, and it allows **access to different elements and properties**. In this code, Log is just one of the elements of Debug, and x is just one element of the position, which is itself just one element of transform, all of which are properties of debugging.

log is an element of debug
x is an element of position properties

For example, "transform.position.x" in this part of the code transform is the country, the position is the city, and x is the street within the city we're trying to locate. So the dot operator is effectively allowing you to separate or access elements of a compound item in Unity, where a compound

item being something that contains many elements.

So, for example, `transform` contains `position`, `rotation`, and `scale`. So, the `dot operator` is used to choose a position, and then the position contains `x`, `y`, or `z`. So we have chosen `x` by using the dot operator once again.

Semi-Colon ;

It is used to `terminate the statements` and will appear at the end of a line of code. However, not everything is a statement, functions, and class declarations do not require a semicolon terminator. It is very easy to forget to add the semicolon.

Indenting

In unity, `indenting is not necessary`, but this will help make your code `a lot more readable` to the human eye. Instead of having all code starting at the left margin, when opening a function, the `code inside the curly brackets should be indented one level`. Likewise, any function inside this should be indented one level more, so you can visually see which code is part of which function.

Comments

`Comments are the part of the code which has no technical function; it is just an explanation or description of the source code.` It helps a developer to `explain the logic of the code`, to enter the details about the author or programmer, and to `improve program readability`. At `run-time` `comment is ignored by the compiler`.

`//` is a single line comment. Simply put `//` at the start of the line, the compiler will know to skip over it when calculating the code.

`/* */` is a multi-line comment. This multi-line comment can also be used to disable chunks of code without deleting them when testing and debugging your code.

← Prev

Next →

Youtube For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

Aptitude

[Logical Reasoning](#)

Reasoning

[Verbal Ability](#)

Verbal Ability

[Interview Questions](#)

Interview Questions

Company
Interview
Questions

Company Questions

Trending Technologies

Artificial
Intelligence

Artificial
Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science
Tutorial

Data Science

Angular 7
Tutorial

Angular 7

Blockchain
Tutorial

Blockchain

Git Tutorial

Git

Machine
Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures
tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer
Network tutorial

Computer Network

Compiler Design
tutorial

Compiler Design

Computer
Organization and
Architecture

Computer
Organization

Discrete
Mathematics
Tutorial

Discrete
Mathematics

Ethical Hacking

Computer
Graphics Tutorial

Software
Engineering

html tutorial

Ethical Hacking

Computer Graphics

Software
Engineering

Web Technology

Cyber Security
tutorial

Cyber Security

Automata
Tutorial

Automata

C Language
tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework
tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

Data Warehouse
Tutorial

Data Warehouse

If Statements

to cover change in circumstances

1

The variables change potentially in many different circumstances. Like **when the level changes**, **when the player changes their position**, and so on. Accordingly, you will often need to check the value of a variable to branch the execution of your scripts that perform **different sets of actions** depending on the value.

For example, if **bikesPetrol** reaches 0 percent, you will **perform a death sequence**, but if **bikesPetrol is at 20 percent**, you might **only display a warning message**.

C# offers two main conditional statements to achieve a program branching like this in Unity. These are the switch statement and if statement.

The **?if statement?** has various forms. The most basic form checks for a condition and will execute a subsequent block of code if and only if the condition is true.

Let's see one simple example:

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

public class IfStatement : MonoBehaviour
{
    public int myNumber = 10;

    // Use this for initialization
    void Start()
    {
        if (myNumber > 5)
        {
            print("myNumber is greater than 5");
        }
    }
}
```



```

        }

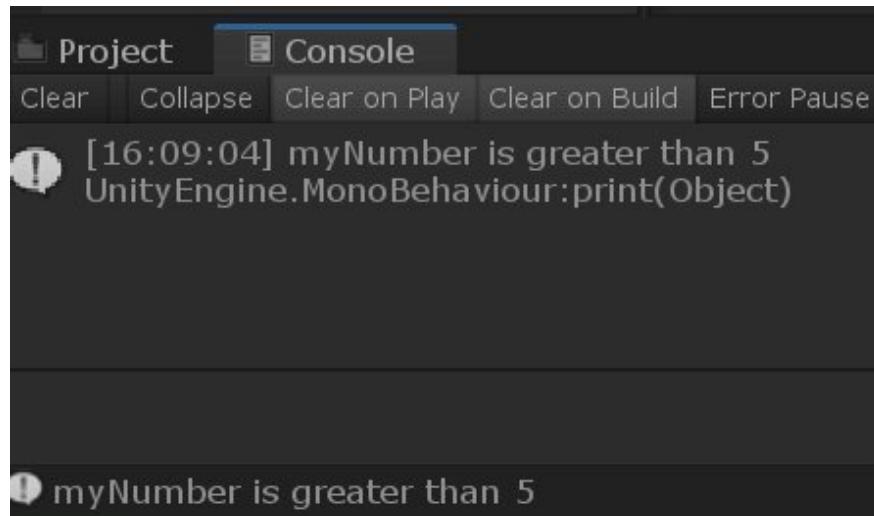
    }

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

}

```

Attach this script file to the GameObject's component. And when you play this project, it will display the following output in console:



If else Statement

Let's see one example for if-else statement:

```

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

public class IfStatement : MonoBehaviour
{
    public int myNumber = 15;

    // Use this for initialization
}

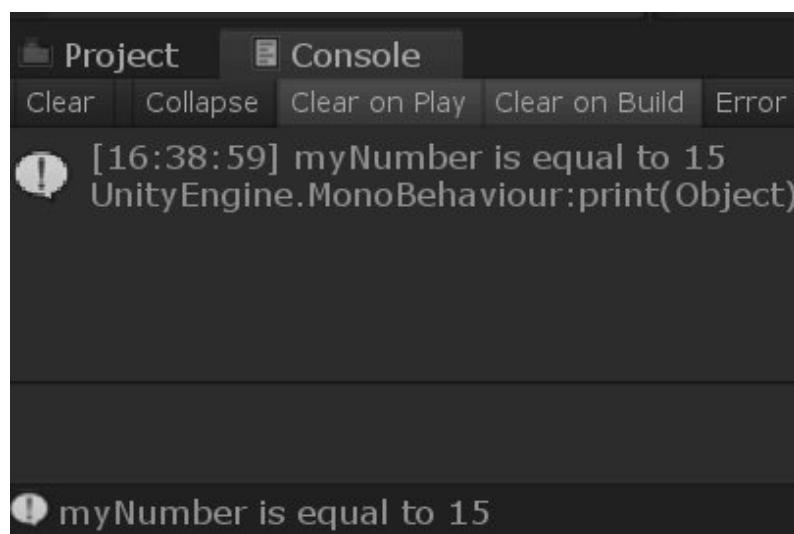
```

```
void Start()
{
    if (myNumber == 10)
    {
        print("myNumber is equal to 10");
    }
    else if (myNumber == 15)
    {
        print("myNumber is equal to 15");
    }
    else
    {
        print("myNumber is not equal to 10");
    }
}

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



Output:



```
Project Console
Clear Collapse Clear on Play Clear on Build Error
[16:38:59] myNumber is equal to 15
UnityEngine.MonoBehaviour:print(Object)

! myNumber is equal to 15
```

← Prev

Next →

Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

Data Structures tutorial

DAA tutorial

Operating System

DBMS

Data Structures

DAA

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Computer Organization

Discrete Mathematics Tutorial

Discrete Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse

Loops

for repeating action

Loops in programming are the way of repeating actions. Let's look at three different kinds of the loop, a For loop, a While loop, and a Do While loop.

While Loop

The ?while loop? is used to perform an action, while a condition is met.

To make a while loop, start with the keyword ?while? followed by brackets. Within the brackets, you must write a condition. Whenever the condition is true, the code within the loop block will be executed:

Syntax:

```
while(condition){  
    //loop block  
}
```



Example:

Let's see an example. In this example, we have a variable called PlayerLives. We have a total four life of a player. While the number of life of a player is more than zero, the player can play the game. Means, this loop will continue four times because the player has four lives. And after that, the Player will lose the game.

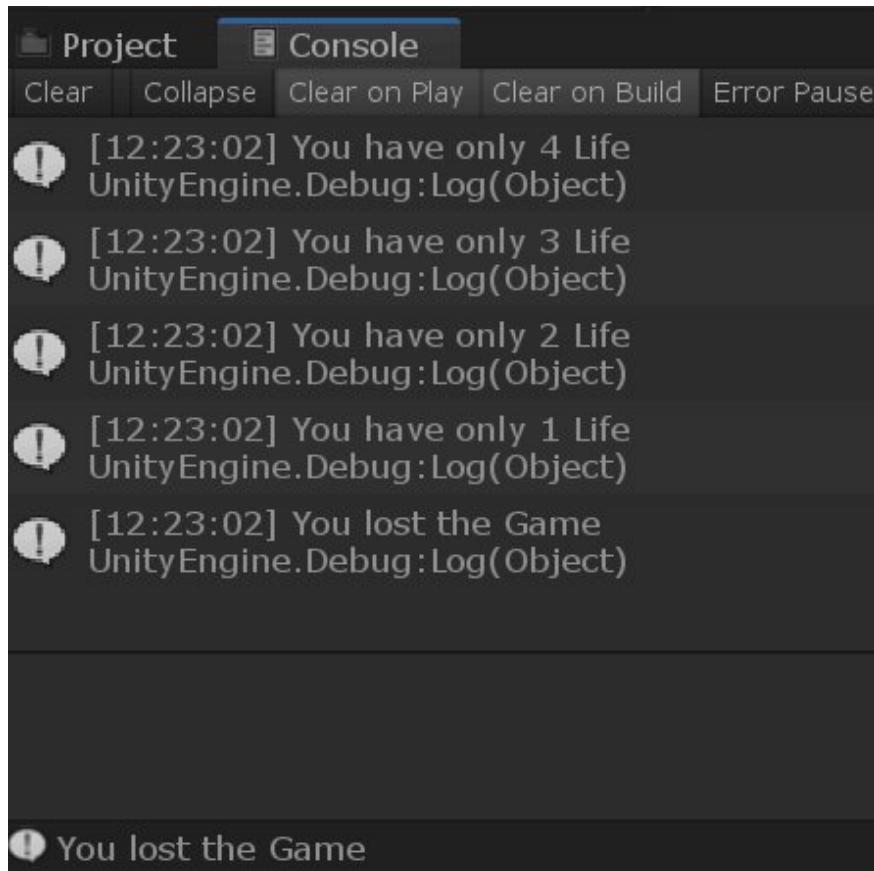
```
using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class WhileLoop : MonoBehaviour  
{
```

```
int PlayerLives = 4;

void Start()
{
    while (PlayerLives > 0)
    {
        Debug.Log("You have only " + PlayerLives + " Life");
        PlayerLives--;
    }
    Debug.Log("You lost the Game");
}
```



Output:



The screenshot shows the Unity Editor's Console window. The tab bar at the top has 'Project' and 'Console' selected. Below the tabs are buttons for 'Clear', 'Collapse', 'Clear on Play', 'Clear on Build', and 'Error Pause'. The console output is as follows:

```
[12:23:02] You have only 4 Life
UnityEngine.Debug:Log(Object)
[12:23:02] You have only 3 Life
UnityEngine.Debug:Log(Object)
[12:23:02] You have only 2 Life
UnityEngine.Debug:Log(Object)
[12:23:02] You have only 1 Life
UnityEngine.Debug:Log(Object)
[12:23:02] You lost the Game
UnityEngine.Debug:Log(Object)
```

At the bottom of the window, a message is highlighted in yellow: 'You lost the Game'.

Do While Loop

The do-while loop is almost identical to the While loop, with one major difference. While loop tests the condition before the loop body, however, do-while loop tests the condition at the end of the

body. This difference means that the body of the do-while loop is guaranteed to run at least once.

Here is the syntax of the do-while loop:

Syntax:

```
do{  
    //body of a loop  
} while(condition);
```

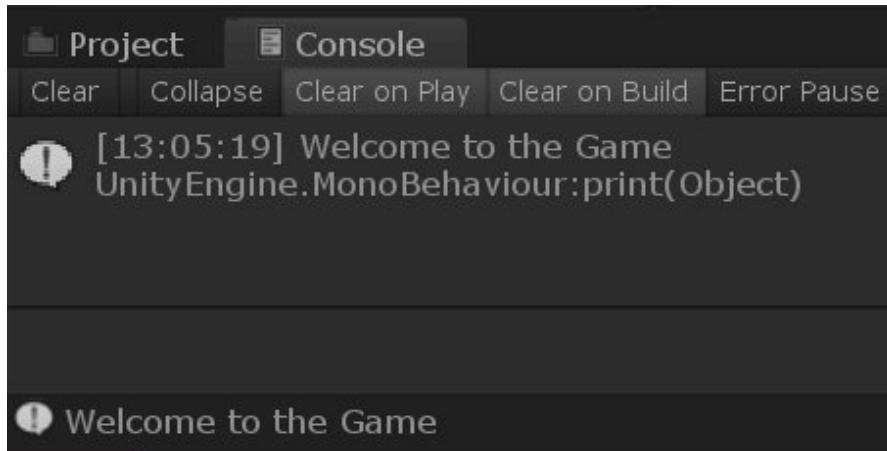
Example:

In this example, we can see that we have a Boolean variable called `shouldContinue`. This variable is set to false. Next, we have the Do while loop. We start with the keyword `do` followed by open and closed braces, whatever the code is between these braces makes up the body of the loop. After the body `?while` keyword? is there followed by the condition. In this case, the loop would only continue while the variable `shouldContinue` is equal to true.

```
using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class DoWhileLoop : MonoBehaviour  
{  
    void Start()  
    {  
        bool shouldContinue = false;  
  
        do  
        {  
            print("Welcome to the Game");  
  
        } while (shouldContinue == true);  
    }  
}
```



Output:



The image shows a screenshot of the Unity Editor's Console window. The title bar has 'Project' and 'Console' tabs. Below the tabs are buttons for 'Clear', 'Collapse', 'Clear on Play', 'Clear on Build', and 'Error Pause'. The main area of the window displays the following text:
! [13:05:19] Welcome to the Game
UnityEngine.MonoBehaviour:print(Object)

Below the main window, there is a smaller, semi-transparent window with the same text: 'Welcome to the Game'.

For Loop

?For Loop? is probably the most common and **flexible loop**. The ?for loop? works by creating a loop with a controllable **number of iterations**. Functionally it begins by checking conditions in the loop. After each loop, known as an iteration, it can optionally increment a value.

The syntax for this has **three arguments**. The first one is iterator; this is used to count through the iterations of the loop. The second argument is the condition that must be true for the loop to continue. Finally, the third argument defines what happens to the iterator in each loop.

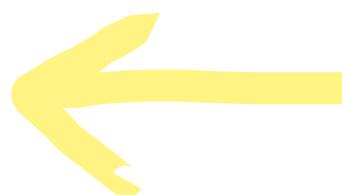
Syntax:

```
for(int i = 0; i<10; i++){  
    //loop block  
}
```



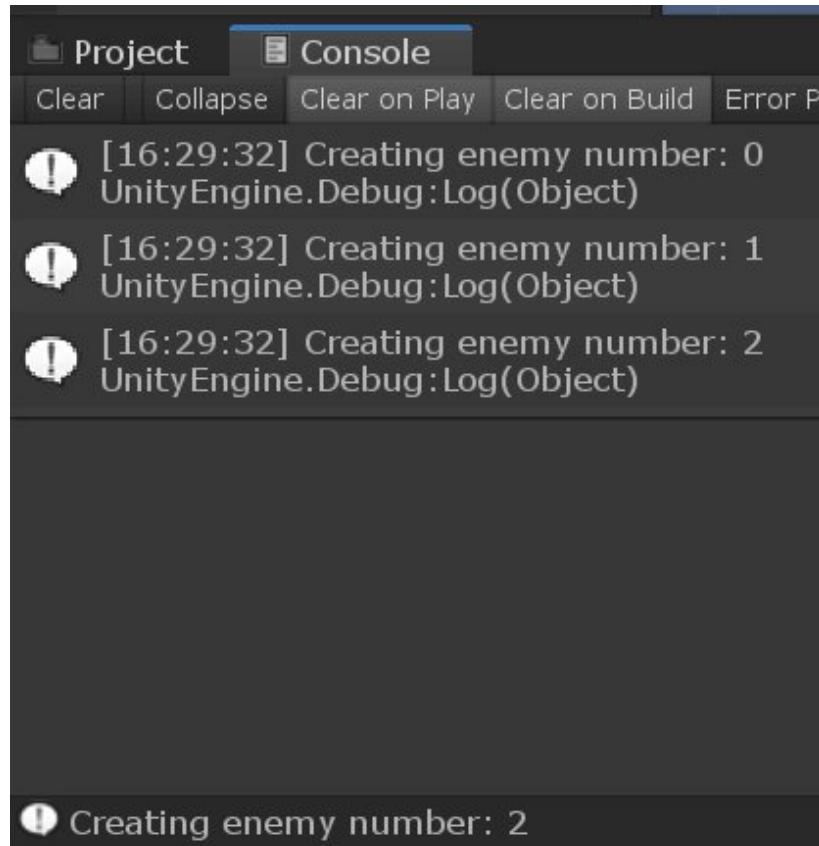
Example:

```
using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class ForLoop : MonoBehaviour  
{  
    int numEnemies = 3;  
  
    void Start()  
    {
```



```
for (int i = 0; i < numEnemies; i++)  
{  
    Debug.Log("Creating enemy number: " + i);  
}  
}  
}
```

Output:



For Each Loop

The **foreach loop** is **very simple and easy to use**. It has the simplest syntax. The **foreach** keyword followed by brackets is used in this loop. You must specify the type of data you want to iterate inside the brackets.

Pick a single element variable name, give a name to this variable whatever you want. This name is used to access this variable inside the main loop block. After the name, write in a keyword, followed by our List variable name.

Syntax:

```
foreach(Type element name in myCollectionVariable){  
    //block of code  
}
```

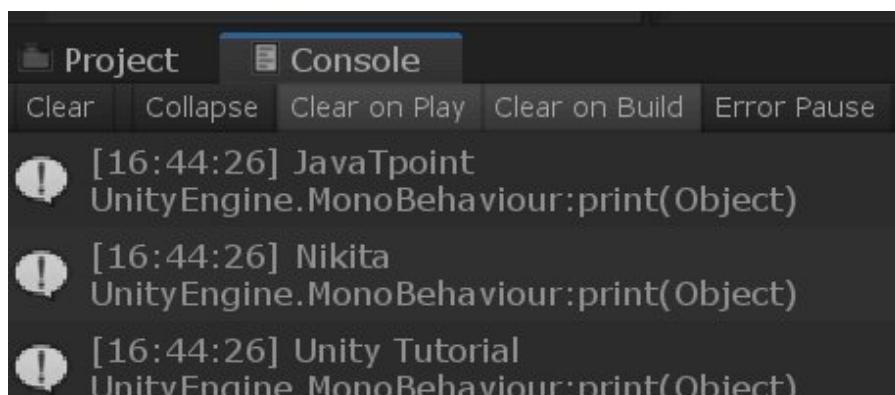


Example:

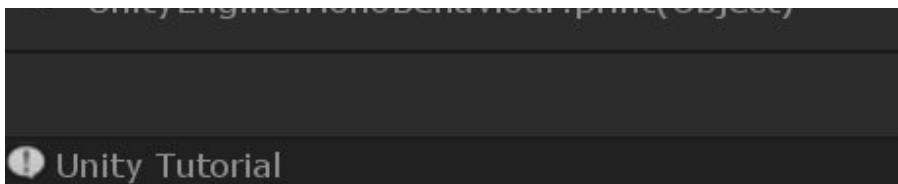
```
using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class ForEachLoop : MonoBehaviour  
{  
    void Start()  
    {  
        string[] names = new string[3];  
  
        names[0] = "JavaTpoint";  
        names[1] = "Nikita";  
        names[2] = "Unity Tutorial";  
  
        foreach (string item in names)  
        {  
            print(item);  
        }  
    }  
}
```



Output:



```
Project Console  
Clear Collapse Clear on Play Clear on Build Error Pause  
! [16:44:26] JavaTpoint  
UnityEngine.MonoBehaviour:print(Object)  
! [16:44:26] Nikita  
UnityEngine.MonoBehaviour:print(Object)  
! [16:44:26] Unity Tutorial  
UnityEngine.MonoBehaviour:print(Object)
```



← Prev

Next →

Youtube For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

Python Pillow
tutorial

Python Pillow

Python Turtle
tutorial

Python Turtle

Keras tutorial

Keras

Preparation

Aptitude

Aptitude

Logical
Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview
Questions

Interview Questions

Company
Interview
Questions

Company Questions

Trending Technologies

Artificial
Intelligence

Artificial
Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science
Tutorial

Data Science

Angular 7
Tutorial

Angular 7

Blockchain
Tutorial

Blockchain

Git Tutorial

Git

Machine
Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial DBMS	Data Structures tutorial Data Structures	DAA tutorial DAA	Operating System Operating System
Computer Network tutorial Computer Network	Compiler Design tutorial Compiler Design	Computer Organization and Architecture Computer Organization	Discrete Mathematics Tutorial Discrete Mathematics
Ethical Hacking Ethical Hacking	Computer Graphics Tutorial Computer Graphics	Software Engineering Software Engineering	html tutorial Web Technology
Cyber Security tutorial Cyber Security	Automata Tutorial Automata	C Language tutorial C Programming	C++ tutorial C++
Java tutorial Java	.Net Framework tutorial .Net	Python tutorial Python	List of Programs Programs
Control Systems tutorial Control System	Data Mining Tutorial Data Mining	Data Warehouse Tutorial Data Warehouse	

Scope and Access Modifiers

The scope of a variable is the area in code that can be used by the variables. A variable is supposed to be local to the place in code that it can be used. Code blocks are generally what defines a variable's scope, and are denoted by the braces.

Access modifiers are the keywords. These modifiers are used to specify the declared accessibility of a member or a type. There are four access modifiers in C#:

- o **public**
- o **protected**
- o **internal**
- o **private**

The following six accessibility levels can be specified in C# using the access modifiers are given below:

public: In this, access is not restricted.

Assignment :write a report about access modifiers in unity c#

protected: Here, access is limited to the class in which it is contained.

internal: In this case, access is limited to the current assembly.

protected internal: For this access is limited to the current assembly.

private: Access is limited to the containing type.

private protected: For this access is limited to the containing class or types derived from the containing class within the current assembly.

Example:

In this example, everything within the class can be said to be local to that class. The variables crayons, pens, and answers are all local to the example function and can't be used outside. You would say that the variables beta, alpha, and gamma are in scope within the scope and access

modifiers class. And you would say that the crayons, pens, and answer variables are in scope within the example function.

MyClass.cs

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

public class MyClass
{
    public int apples;
    public int bananas;

    private int stapler;
    private int sellotape;

    public void FruitMachine(int a, int b)
    {
        int answer;
        answer = a + b;
        Debug.Log("Fruit total: " + answer);
    }

    private void OfficeSort(int a, int b)
    {
        int answer;
        answer = a + b;
        Debug.Log("Office Supplies total: " + answer);
    }
}
```



AccessModifiers.cs

```
using System.Collections;
```

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

public class AccessModifiers : MonoBehaviour
{
    public int alpha = 5;

    private int beta = 0;
    private int gamma = 5;

    private MyClass myOtherClass;

    void Start()
    {
        alpha = 29;

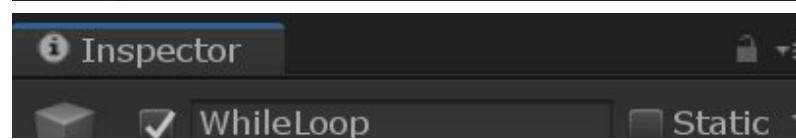
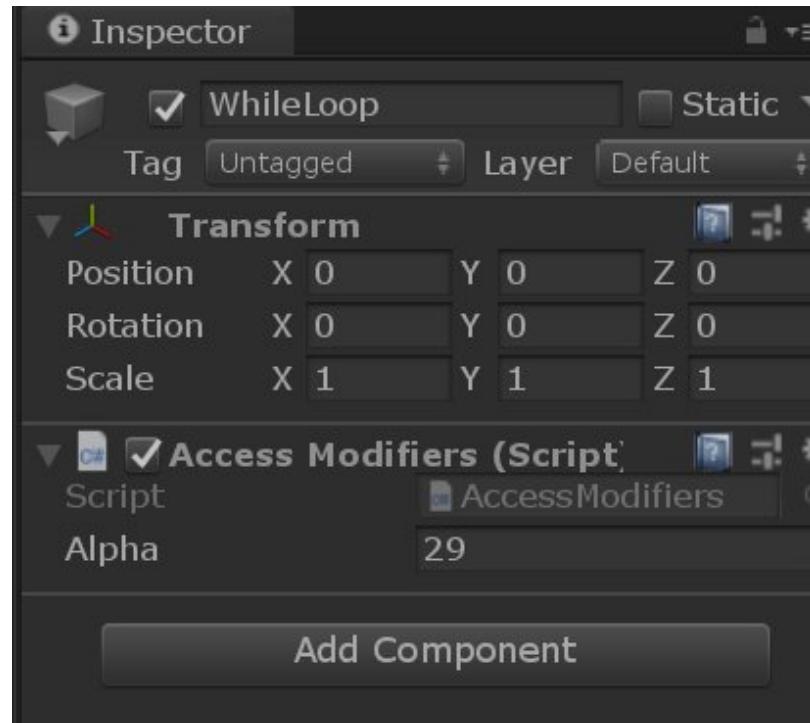
        myOtherClass = new MyClass();
        myOtherClass.FruitMachine(alpha, myOtherClass.appleCount);
    }
}

void Example(int pens, int crayons)
{
    int answer;
    answer = pens * crayons * alpha;
    Debug.Log(answer);
}

void Update()
{
    Debug.Log("Alpha is set to: " + alpha);
}
```

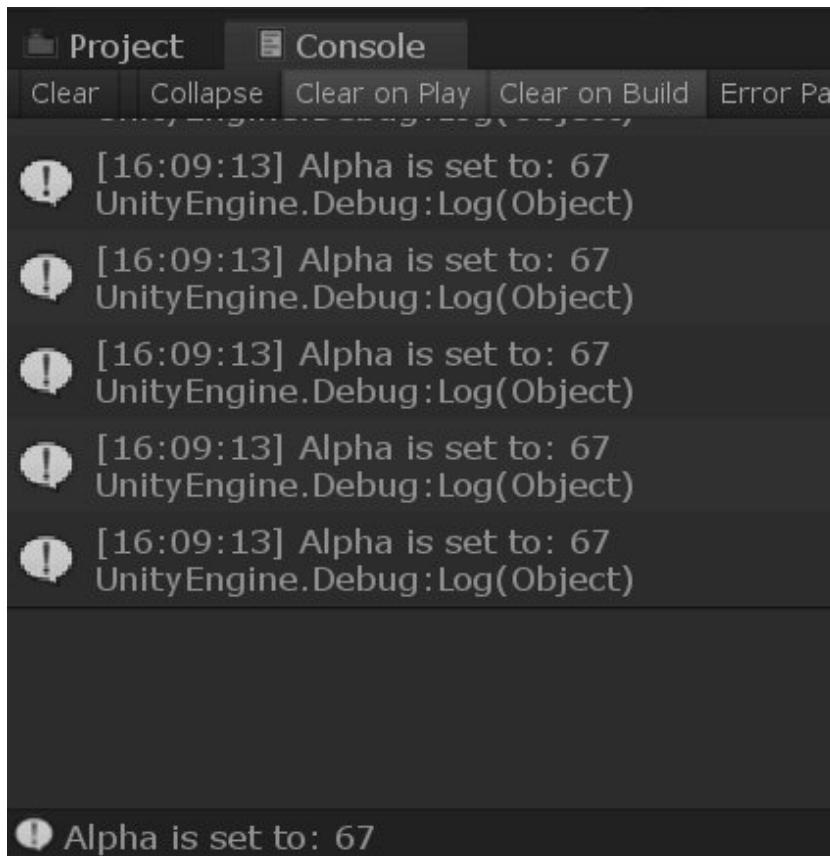
Output:

When you run the game, then you can see that public variable alpha is included as a property that you can edit. This allows the users to edit the variables while they run the game. Imagine, for example, the value controls the speed of a car, and it would be nice to be able to tweak that variable while testing it without having to stop. As such, it makes sense to have this be a public variable.





Here, I changed the value of the Alpha variable to 67 at the run time, and then the output is:



← Prev

Next →

Youtube For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

Aptitude

[Logical Reasoning](#)

Reasoning

[Verbal Ability](#)

Verbal Ability

[Interview Questions](#)

Interview Questions

Company
Interview
Questions

Company Questions

Trending Technologies

Artificial
Intelligence

Artificial
Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science
Tutorial

Data Science

Angular 7
Tutorial

Angular 7

Blockchain
Tutorial

Blockchain

Git Tutorial

Git

Machine
Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures
tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer
Network tutorial

Computer Network

Compiler Design
tutorial

Compiler Design

Computer
Organization and
Architecture

Computer
Organization

Discrete
Mathematics
Tutorial

Discrete
Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse

Data Types

A data type **classifies various types of data**, for example, **string**, **integer**, **boolean**, **float**, and the types of accepted values for that data type, operations that can be performed on the data type, the meaning of the data of that type can be stored.

int: This data type is used **to store 32 bits integer numbers**. This data type stores positive or negative whole numbers.

float: The float data type is used to **store floating point numbers**. The float data type is the default type of number in Unity.

For Example:

```
float points = 72.24;
```

double: Double data type also stores floating point numbers, but it can hold **larger size numbers** than **float**. And it is **not the default number type in Unity**.

For Example:

```
double amount = 134434.23;
```

bool: The bool data type is the short form of Boolean. This data type store true or false values.

For example:

```
bool chance = false;
```

char: This data type stores a **single character** such as a number, letter, space, or special character. A char value should be written in single quotes.

For example:

```
char press = 'a';
```

string: This data type is used to store letters, numbers, and other special characters in the form of words or sentences. A string value should be written inside double quotes.

For example:

```
string CharacterName = "Ken Fighter";
```

Example

Let's see one simple example:

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

public class DataType: MonoBehaviour
{
    int playerHealth = 100;
    float playerVelocity = 14.74f;
    string msg = "Welcome to the Game";
    bool isGameOver = false;

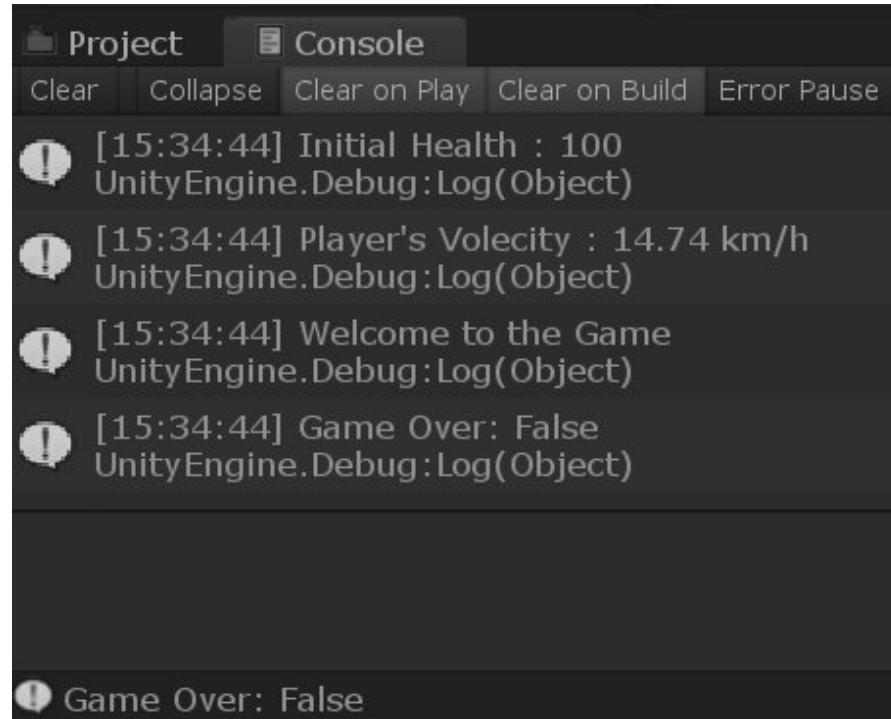
    void Start()
    {
        Debug.Log("Initial Health : " + playerHealth);
        Debug.Log("Player's Volecity : " + playerVelocity + " km/h");
        Debug.Log(msg);
        Debug.Log("Game Over: " + isGameOver);
    }

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



```
{  
}  
}  
}
```

Output:



← Prev

Next →

Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

Aptitude

[Logical Reasoning](#)

Reasoning

[Verbal Ability](#)

Verbal Ability

[Interview Questions](#)

Interview Questions

[Company Interview Questions](#)

Company Questions

Trending Technologies

Artificial Intelligence	AWS Tutorial	Selenium tutorial	Cloud Computing
Artificial Intelligence	AWS	Selenium	Cloud Computing
Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular 7 Tutorial
Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial
Cyber Security	Automata	C Programming	C++

Java tutorial

Java

.Net Framework
tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

Data Warehouse
Tutorial

Data Warehouse

Classes

Classes are the **blueprints for your objects**. Basically, in Unity, **all of your scripts will begin with a class declaration**. Unity automatically puts this in the script for you when you create a new C# script. This class shares the name as the script file that it is in. This is very important because if you change the name of one, you need to change the name of the other. So, try to name your script sensibly when you create it.

The **class** is a **container for variables and functions** and provides, among other things. Class is a nice way to group things that work together.

They are an **organizational tool**, something known as **object-oriented programming** or **OOP** for short. One of the principles of object-oriented programming is to **split your scripts up into multiple scripts**, each one taking a single role or responsibility. Classes should, therefore, be dedicated ideally to one task.

The main **aim of object-oriented programming** is to allow the programmer to develop software in **modules**. This is accomplished through objects. Objects contain data, like integers or lists, and functions, which are usually called methods.

Example

Player.cs

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

public class Player
{
    public string name;
    public int score;
    public int speed;
```

```
public void gameData()
{
    Debug.Log("Player name = " + name);
    Debug.Log("Player power = " + score);
    Debug.Log("Player speed = " + speed);
}
```



PlayerDetails.cs

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

public class PlayerDetails : MonoBehaviour
{
    private Player P1;
    private Player P2;
    private Player P3;

    void Start()
    {
        P1 = new Player();
        P2 = new Player();
        P3 = new Player();

        P1.name = "Bill";
        P1.score = 10;
        P1.speed = 30;

        P2.name = "Bob";
        P2.score = 100;
        P2.speed = 3;

        P3.name = "Jerry";
    }
}
```



```
P3.score = 50;  
P3.speed = 10;  
  
P1.gameData();  
P2.gameData();  
P3.gameData();  
}  
}
```

Output:

Attach the PlayerDetails.cs script file to the GameObject's component and play the game. It will display the following output:



Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Computer Organization

Discrete Mathematics Tutorial

Discrete Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse

Arrays

An array is used to store a sequential collection of values of the same type. In short, an array is used to store lists of values in a single variable. Suppose if you want to store a number of player names, rather than storing them individually as string p1, string p2, string p3, etc. we can store them in an array.

Arrays are a way of storing a collection of data of the same type together.

Declaring an array

To declare an array in C#, you must first say what type of data will be stored in the array. After the type, specify an open square bracket and then immediately a closed square bracket, []. This will make the variable an actual array. We also need to specify the size of the array. It simply means how many places are there in our variable to be accessed.

Syntax:

```
accessModifier datatype[] arrayname = new datatype[array]
```

Example:

```
public string[] name = new string[4];
```



To allocate empty values to all places in the array, simply write the "new" keyword followed by the type, an open square bracket, a number describing the size of the array, and then a closed square bracket.

Example

Let's see one simple example using an array:

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

public class ArrayExample : MonoBehaviour
{
    public int[] playerNumber= new int[5];

    void Start()
    {
        for (int i = 1; i < playerNumber.Length; i++)
        {
            playerNumber[i] = i;
            Debug.Log("Player Number: " +i.ToString());
        }
    }
}
```



Output:



```
[18:59:15] Player Number: 1
UnityEngine.Debug:Log(Object)
[18:59:15] Player Number: 2
UnityEngine.Debug:Log(Object)
[18:59:15] Player Number: 3
UnityEngine.Debug:Log(Object)
[18:59:15] Player Number: 4
UnityEngine.Debug:Log(Object)
Player Number: 4
```

← Prev

Next →

Youtube For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Computer Organization

Discrete Mathematics Tutorial

Discrete Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse

Understanding Collisions

:occurs when a moving object crashes into something

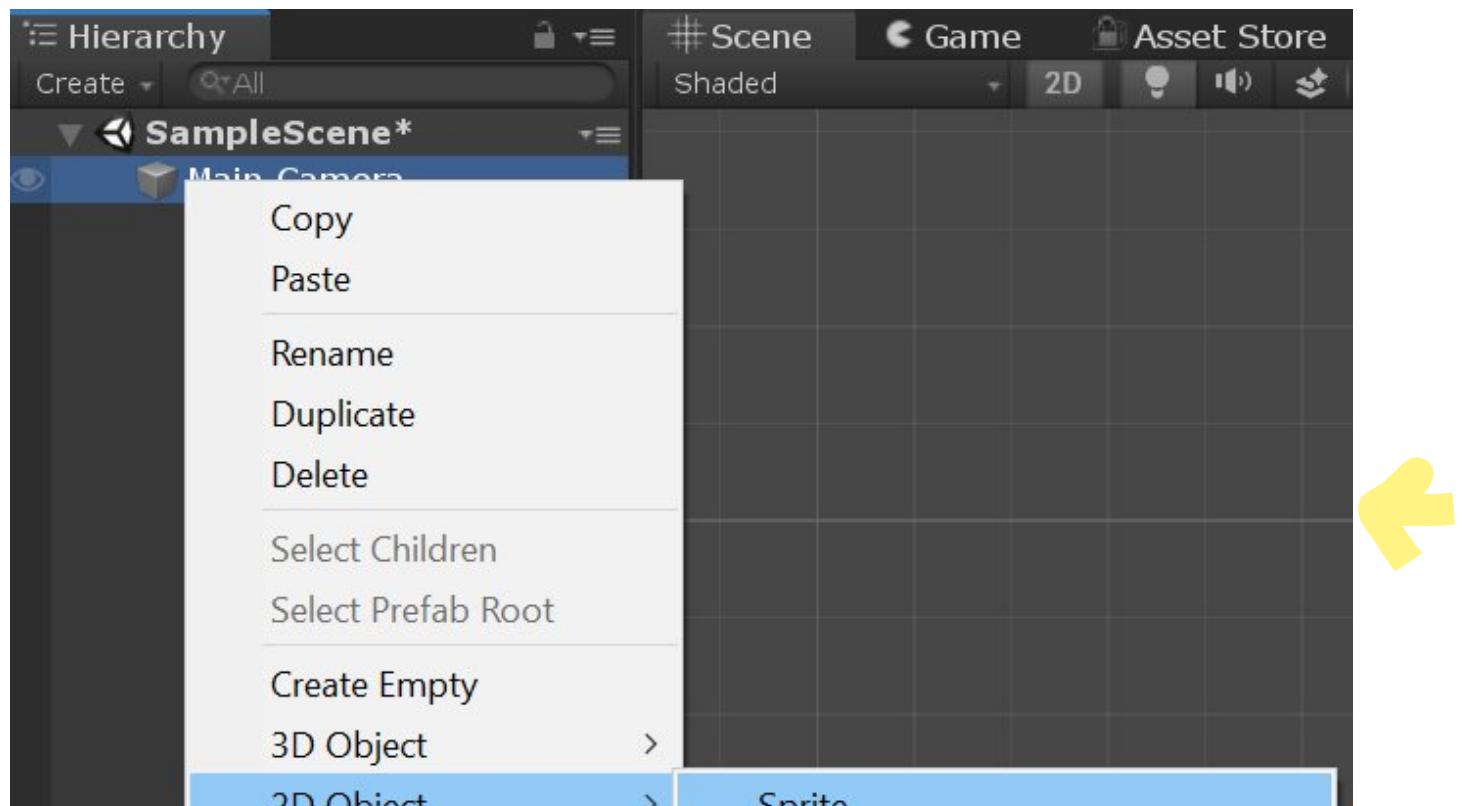
As you know, **everything in your game is a GameObject**. Even the individual tiles that make up your level are GameObjects by themselves.

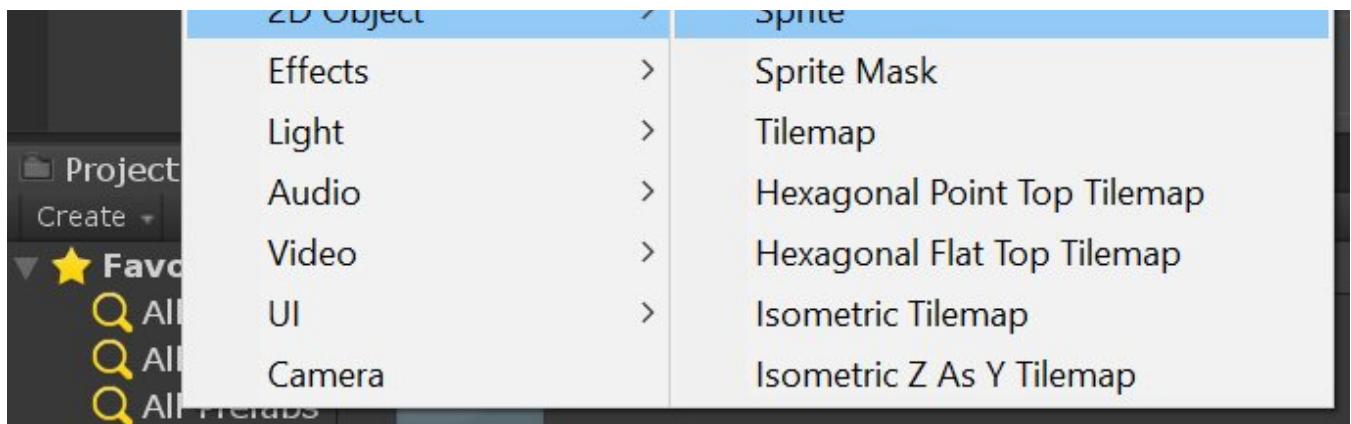
When we consider every component as a GameObject, we understand that there could be **thousands of GameObjects in a scene**, interacting with each other in some way. You can visualize that if unity added collisions to every single GameObject, it would be impractical for the engine to calculate collisions for every single one of them.

In Unity, collisions are separated from the actual sprite itself, attached as separate components, and are calculated on their own.

Collider components describe the shape of an object for the purposes of physical collisions. A **collider will be invisible** and is required to be the exact same shape as the GameObject's mesh.

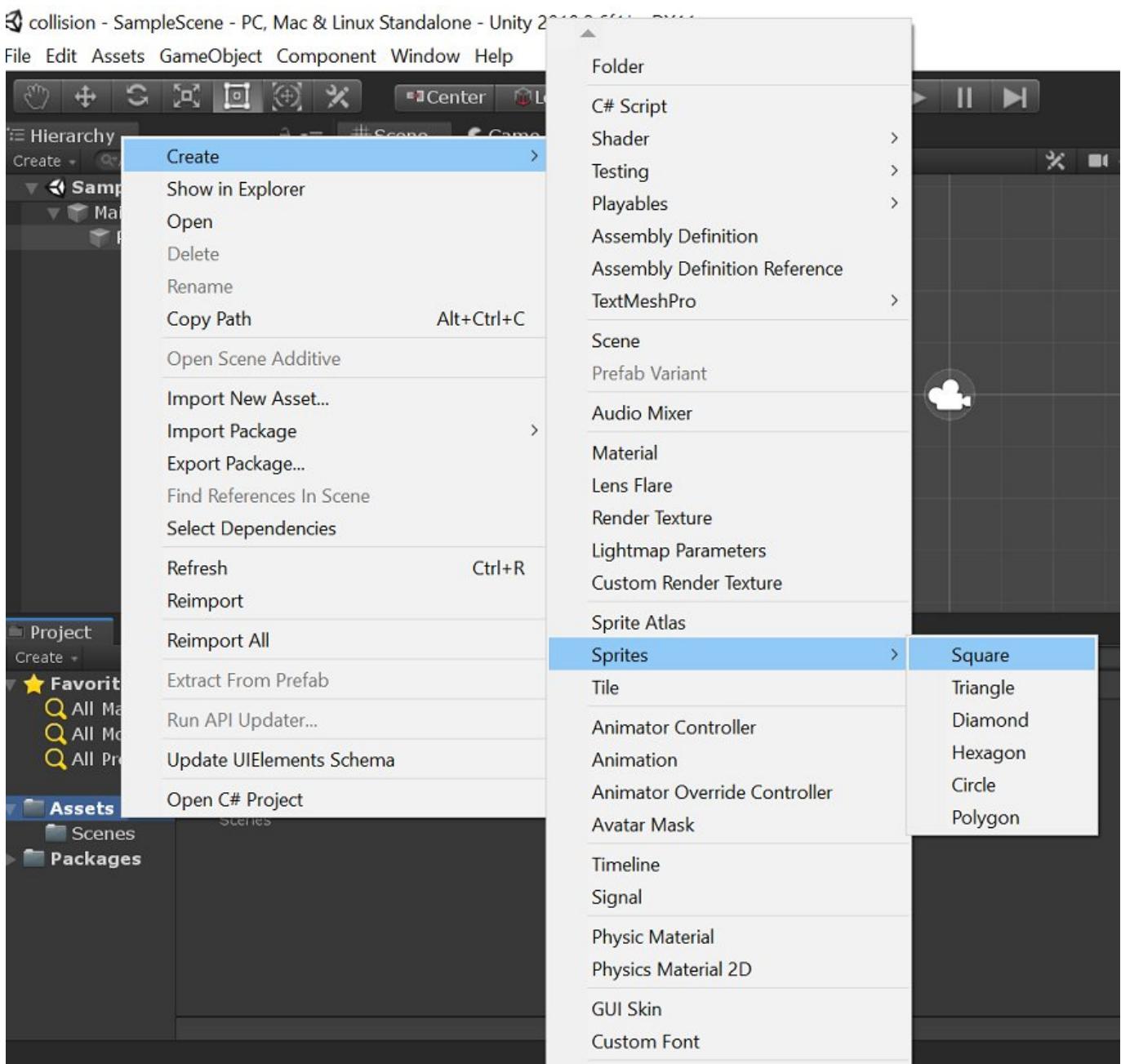
Let's add a simple wall that our player can collide against. First of all, create a sprite. To do that, right click on your scene from the Hierarchy tab and select **2D object -> Sprite**.





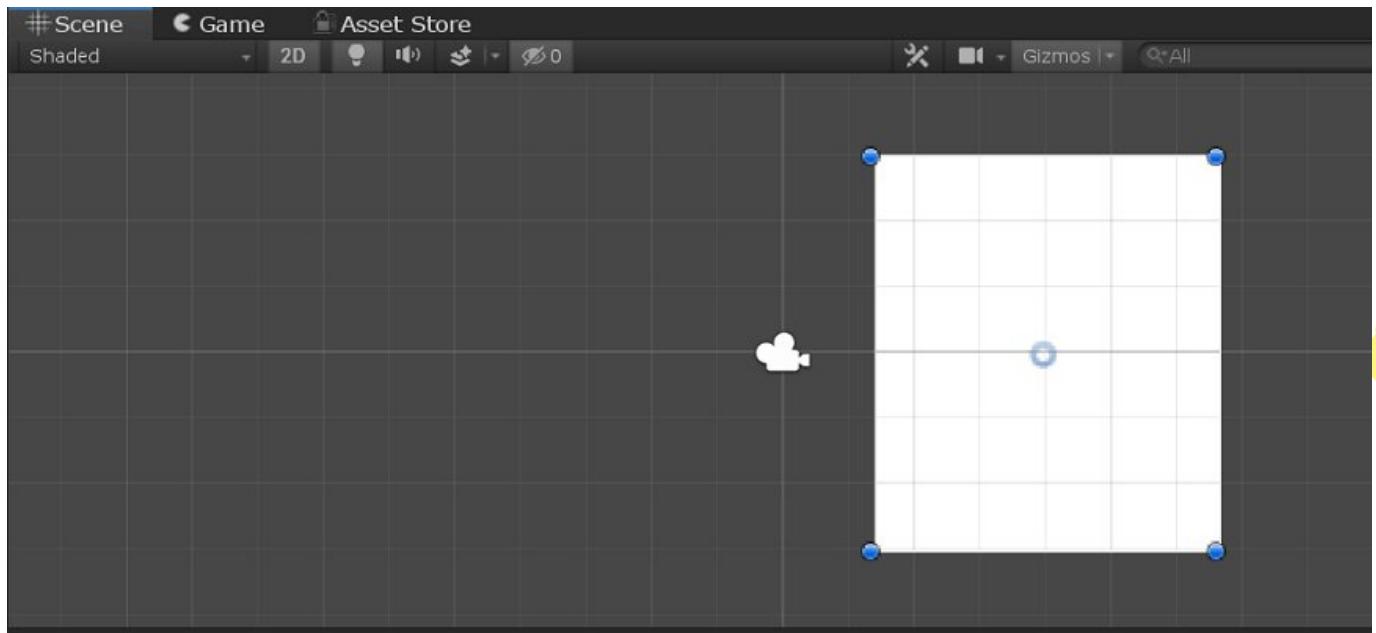
Rename the New Sprite to Player.

Or directly go to the project tab and right click on Assets and select Create -> Sprites -> Square.

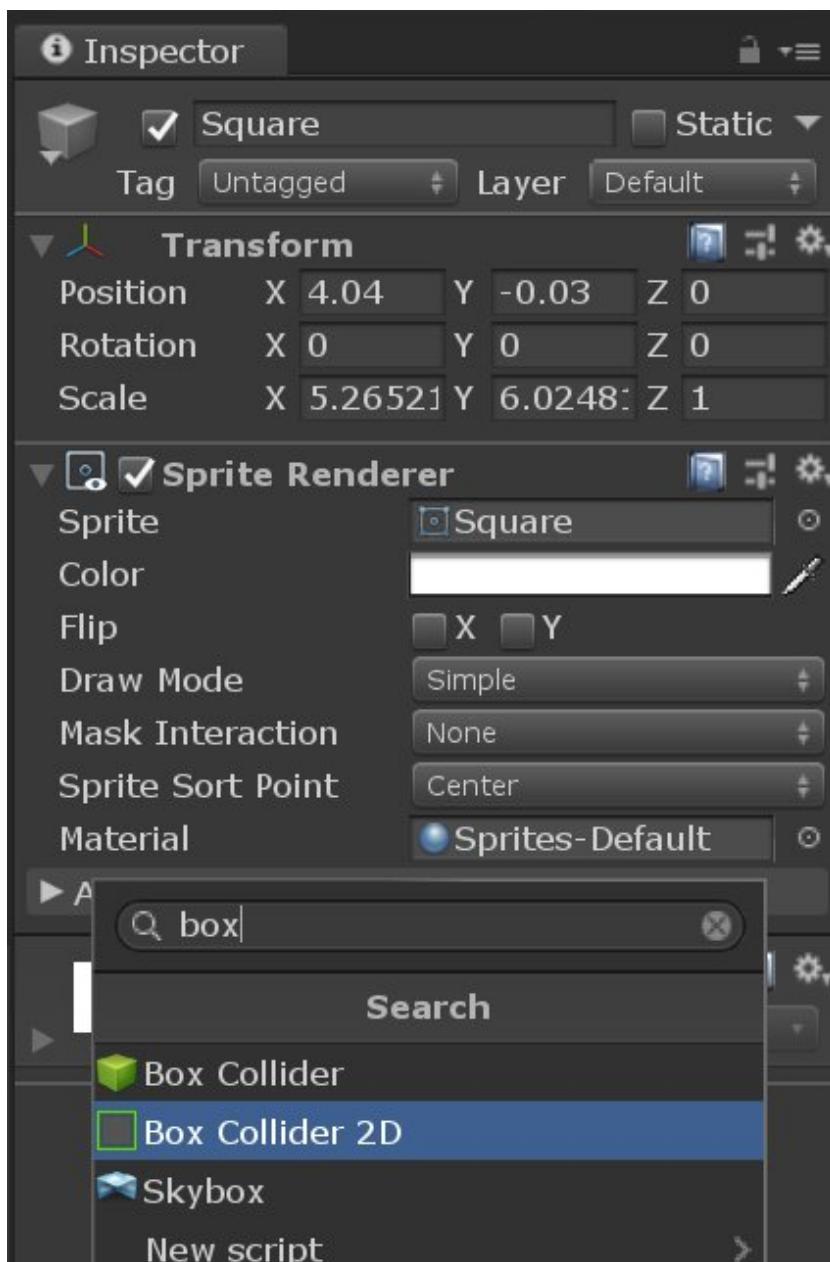


Drag that square to the scene.



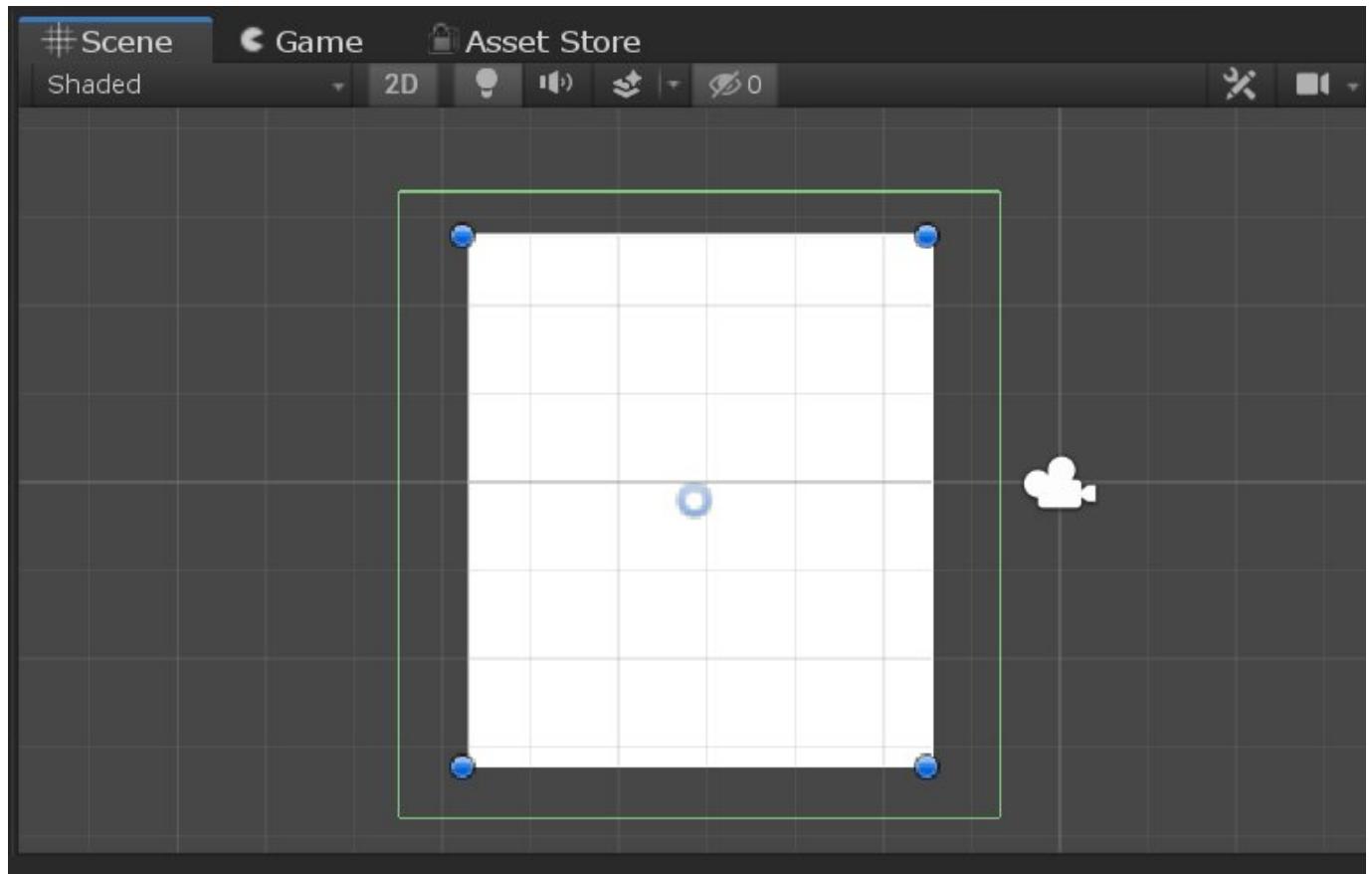


Now go to **Add Component** in the **Inspector** tab, and search for "Box Collider 2D".





You will see a bright green line on the perimeter of your Player GameObject. This is the **collision boundary**. It defines the actual shape of the collidable object.



We can create collisions of different shapes in different sizes. They can be in a rectangular shape, or even they can be in polygonal shapes.

← Prev

Next →

Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

Aptitude

[Logical Reasoning](#)

Reasoning

[Verbal Ability](#)

Verbal Ability

[Interview Questions](#)

Interview Questions

[Company Interview Questions](#)

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Computer Organization

Discrete Mathematics Tutorial

Discrete Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software

html tutorial

Web Technology

Engineering

Cyber Security
tutorial

Cyber Security

Automata
Tutorial

Automata

C Language
tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework
tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

Data Warehouse
Tutorial

Data Warehouse

Understanding Prefabs and instantiation

prefabs having a template instead of creating object from scratch

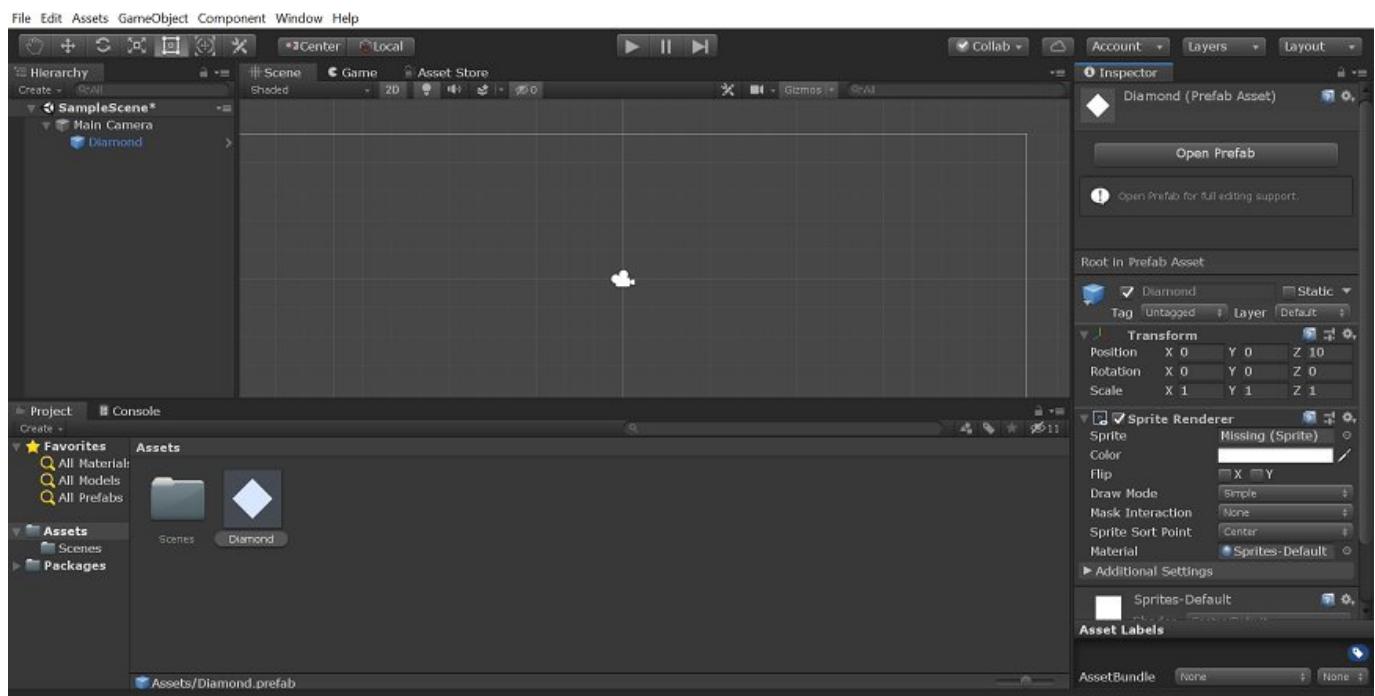
Instantiating and destroying characters or objects is very common in any game. Instantiating means bringing the object into existence. Objects appear or spawn or generate in a game, enemies die, GUI elements vanish, and scenes are loaded all the time in the game.

Prefabs are very useful when you want to instantiate complicated GameObjects or collection of GameObjects at run time. Compared with creating GameObjects from scratch using code, instantiating prefabs using code is better, and it has many advantages. instantiation :bringing object into existence

Let's understand what prefabs are; since prefabs are considered important to understand how instantiation works in Unity.

Prefabs are like blueprints of a GameObject. So we can say, Prefabs are a copy of a GameObject that can be duplicated and put into a scene, even if it didn't exist when the scene was being made; in other words, prefabs can be used to generate GameObjects dynamically.

Let's create a prefab; for this, you need to drag the desired GameObject from your scene hierarchy into the project Assets.



Now, to instantiate a GameObject, we call the `Instantiate()` method in our script. This method is available in `MonoBehaviour`, takes in a `GameObject` as a parameter, so it knows which `GameObject` to create or duplicate. It also has different overrides for changing the newly instantiated object's transform, as well as parenting.

Now create a new script called `Instantiator.cs` and copy the following code in it.

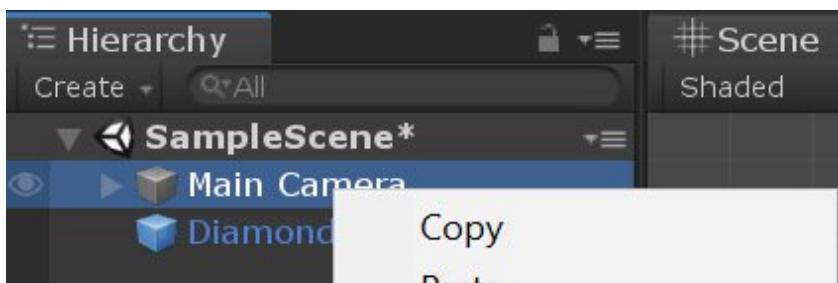
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

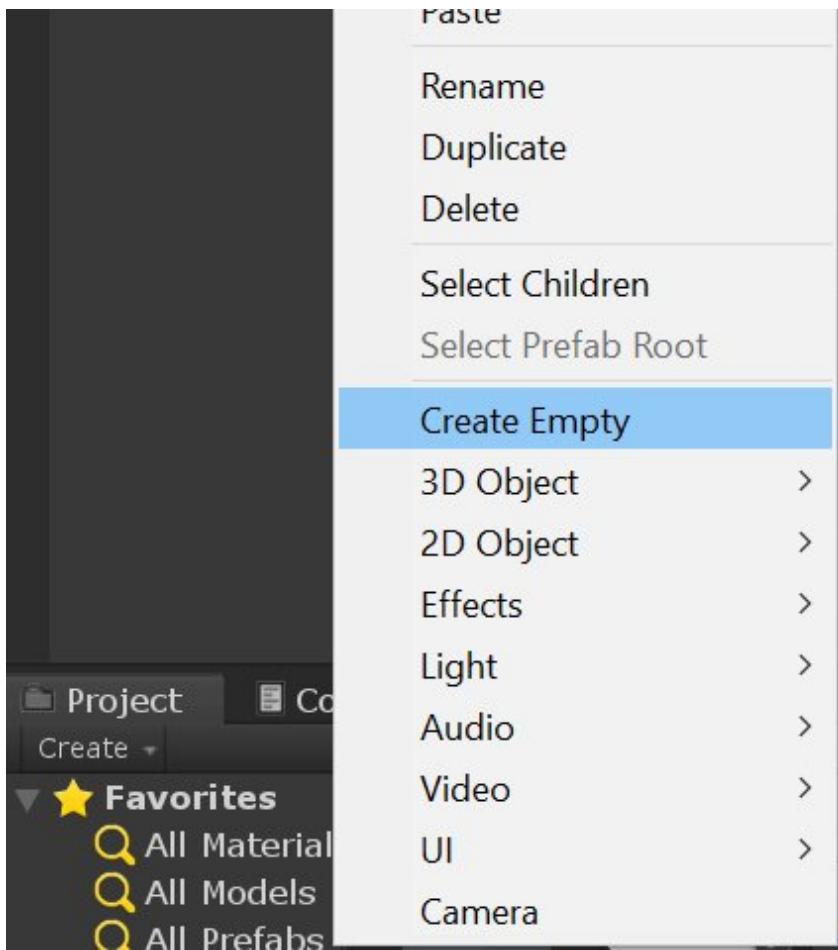
public class Instantiator : MonoBehaviour
{
    public GameObject Diamond;
    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            Instantiate(Diamond);
        }
    }
}
```

let it be alive and work

In the above program, we used the `GetKeyDown` method of the `Input` class to check if the player pressed a specific button during the last frame. The `GetKeyDown()` method returns true if the key specified by the `KeyCode` enum (which is used to list all possible keys on a standard keyboard) is pressed in that frame.

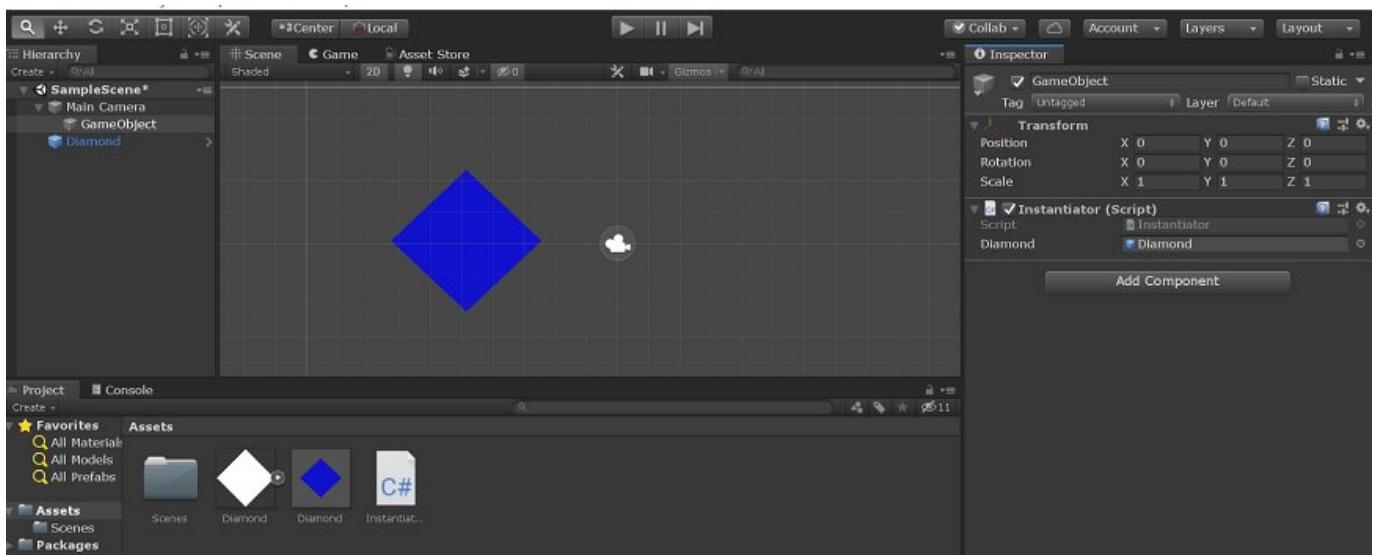
Save the script. And now, create an empty `GameObject` by right-clicking on the scene from the `Hierarchy` tab.





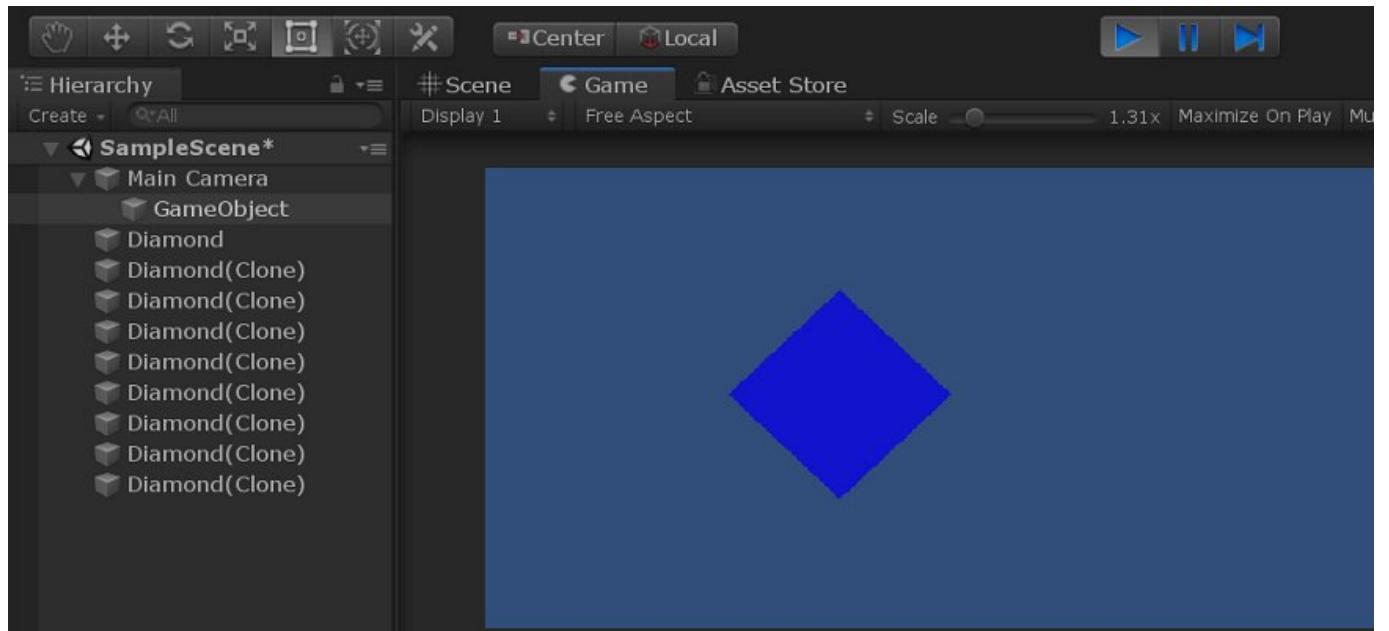
Attach that script file (Instantiator.cs) to the newly created GameObject's Component from the Inspector tab.

Now, in the Diamond variable drag that prefab we have created.



When you run the game now, pressing the Spacebar will create a new Diamond object identical to the one we used to create the prefab. You can see each diamond is created in the object hierarchy. The reason you can't see them show up in the game is that for the time being, they are all being created exactly one over the other. But you can verify it from the Hierarchy tab. Each

time when you press a space bar, it will show in the tab as a Diamond (clone).



← Prev

Next →

[Youtube](#) For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

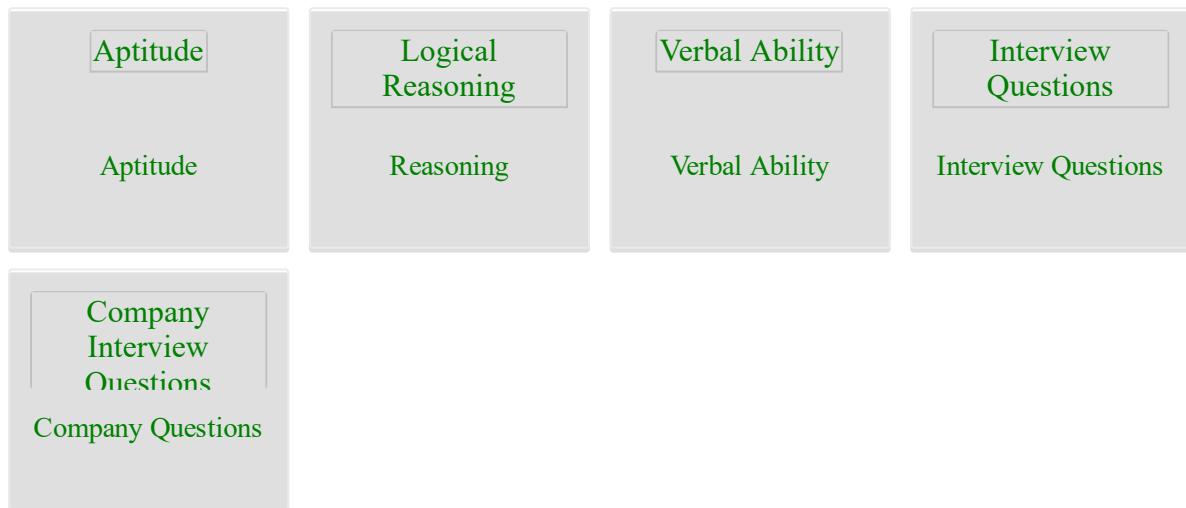
Swagger

[T-SQL tutorial](#)

Transact-SQL



Preparation



Trending Technologies



Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular 7 Tutorial
Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial
Cyber Security	Automata	C Programming	C++
Java tutorial	.Net Framework tutorial	Python tutorial	List of Programs
Java	.Net	Python	Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

Data Warehouse
Tutorial

Data Warehouse

Unity GameObject Destruction

Like instantiation, the destruction of GameObjects is also important. In this section, we will understand how to destroy the GameObjects.

Destroying a GameObject is very simple as creating a GameObject. You require a reference to the object to be destroyed, and call the `Destroy()` method with this reference as a parameter.

Let's create 5 diamonds, which destroy themselves when an assigned key is pressed.

First of all, create a new script. Rename it to `Destroyer.cs` and open it on an editor. Now, copy the following code:

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

public class Destroyer : MonoBehaviour
{
    public KeyCode keyToDelete;

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

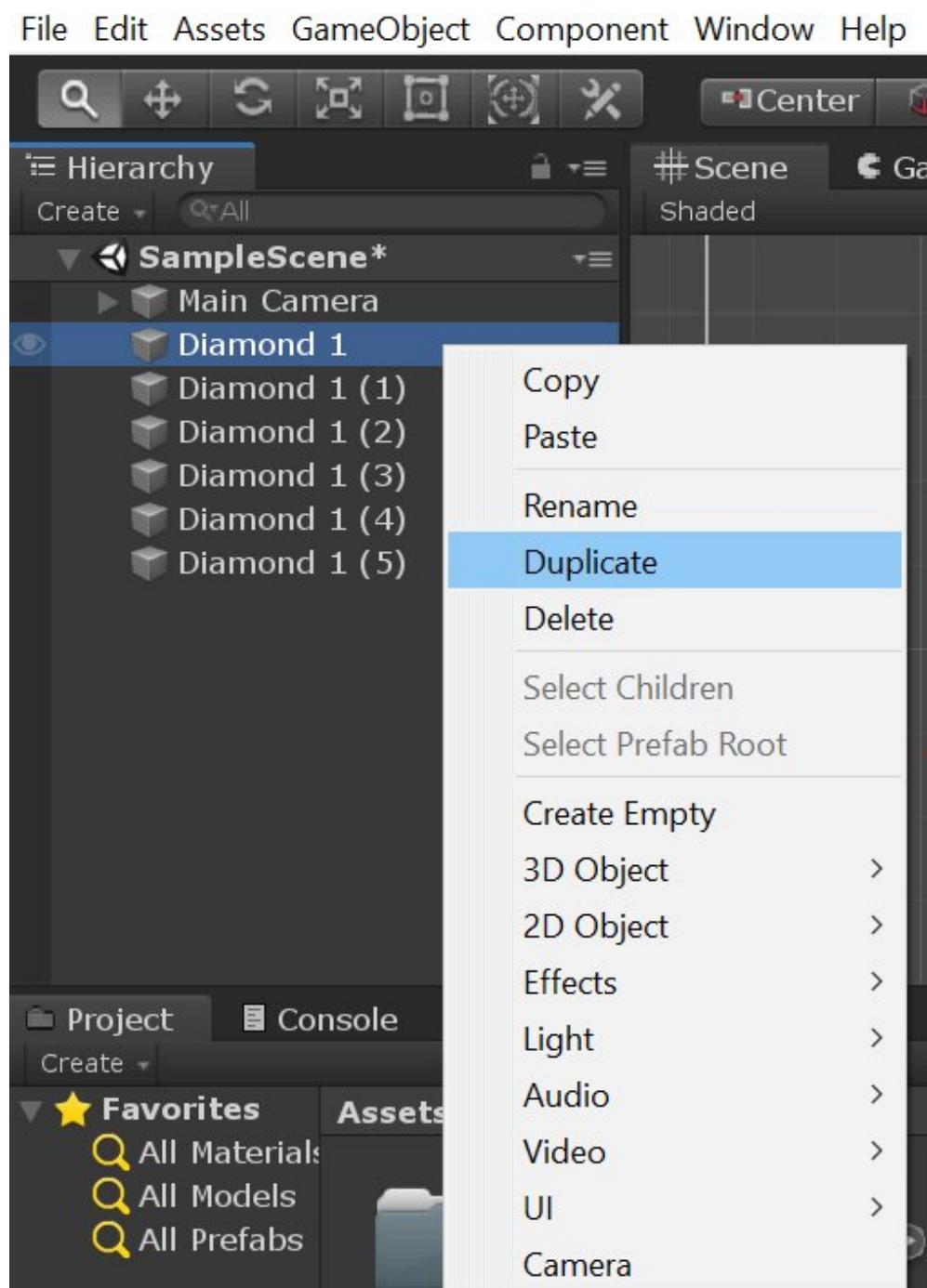
        if (Input.GetKeyDown(keyToDelete)) {
            Destroy (gameObject);
        }
    }
}
```

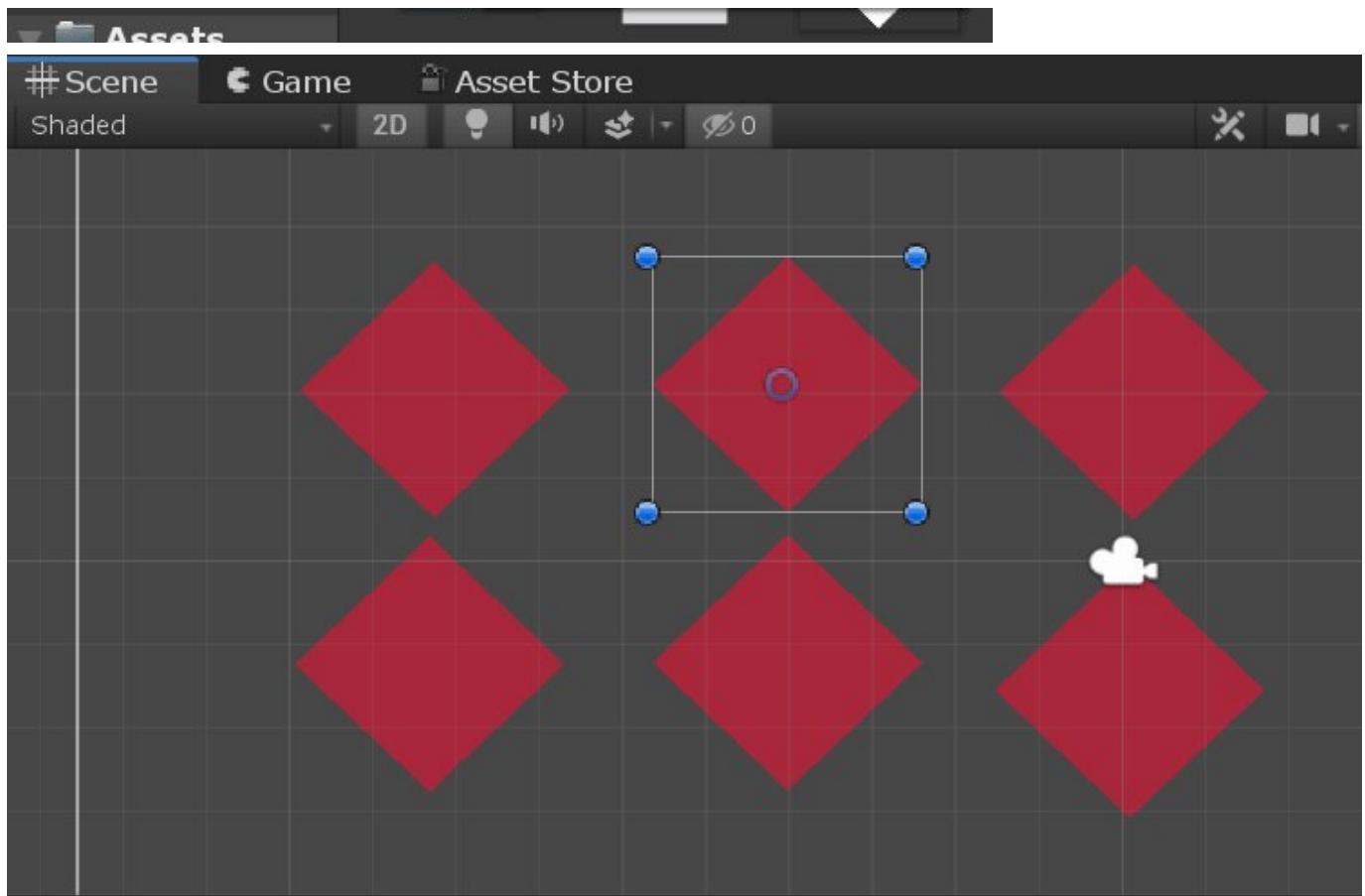
Here, we used a public variable for `KeyCode`. A `KeyCode` is used to describe a key on a standard

keyboard, and the input class in its method uses it. By making this variable public, we can make it accessible through the editor. When the variable is made public, we don't need hardcode values such as KeyCode.A into the code.

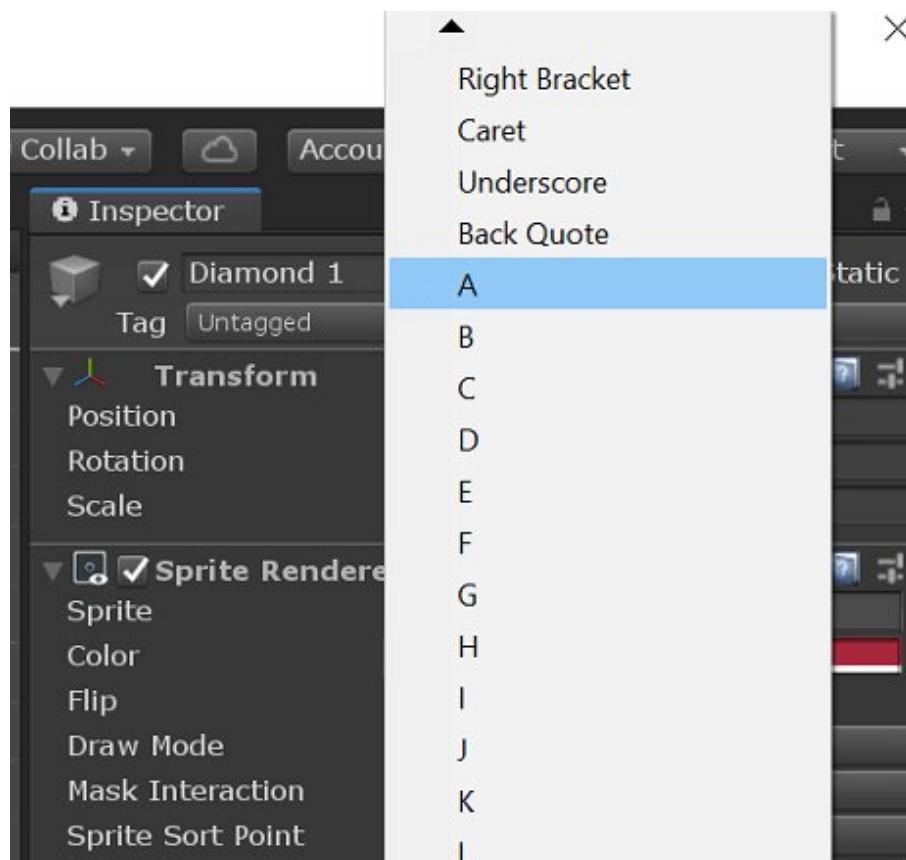
Here, we have added the gameObject variable. This new gameObject variable is used to refer to the gameObject this script is attached to. If you add this script on multiple objects, they will all react the same way whenever this variable is involved.

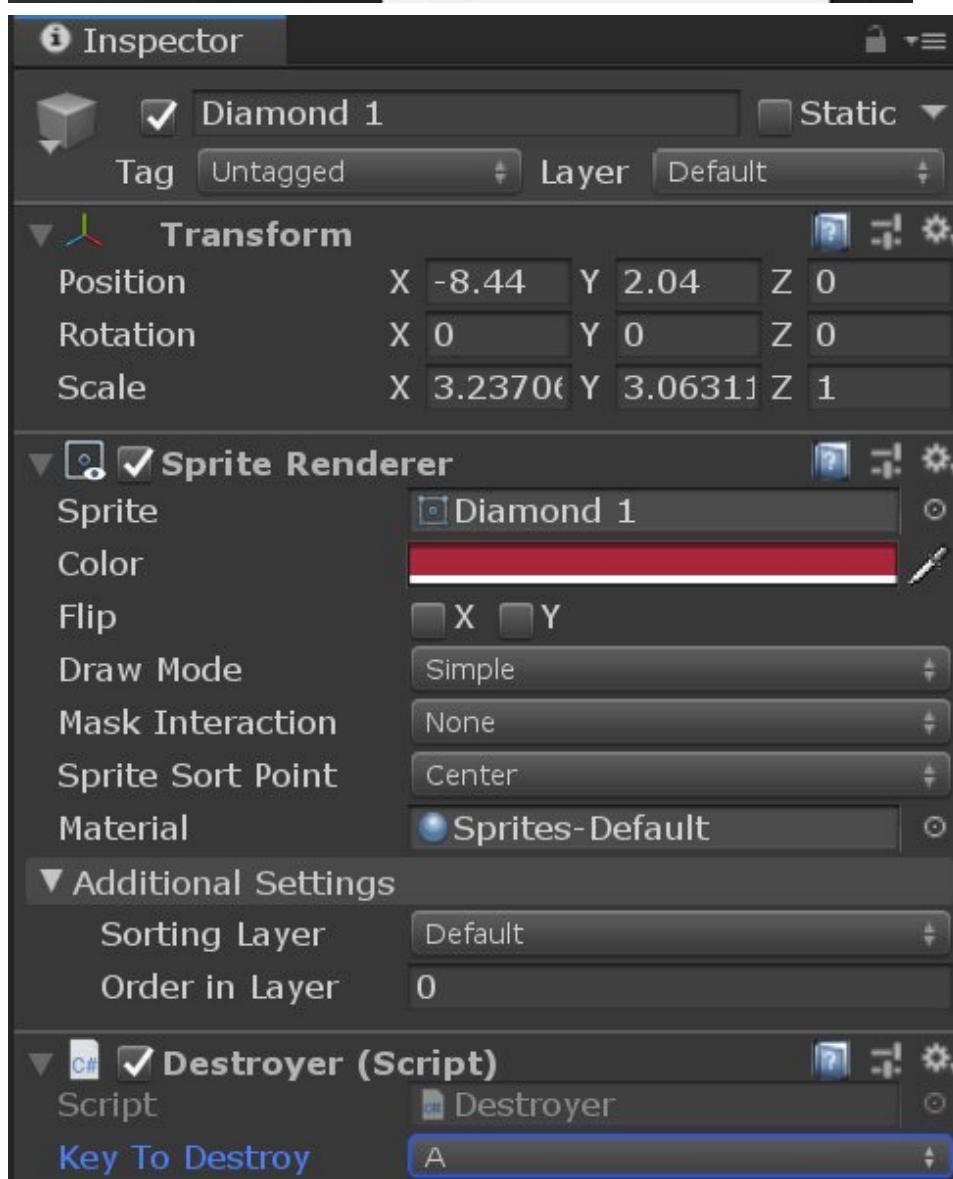
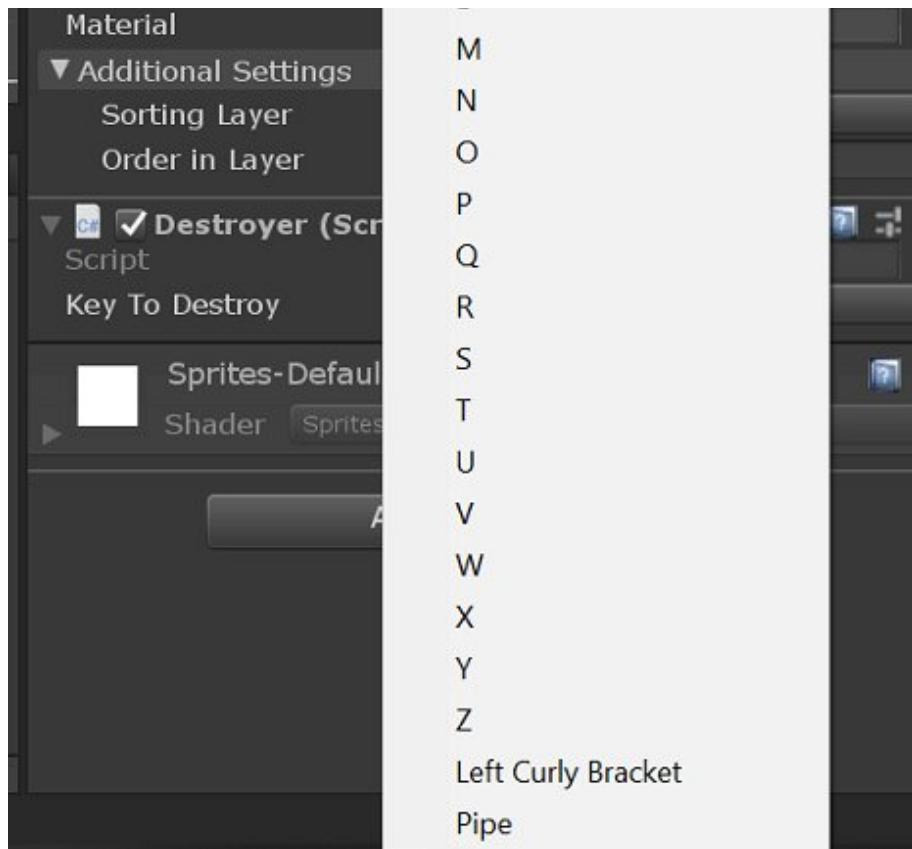
Now, we will create a new diamond sprite and attach our script (Destructor.cs) to it. Next, right-click on the gameObject from the hierarchy tab and select the Duplicate option. A new sprite is produced in the hierarchy tab; you should use the Move tool to reposition it. Repeat the steps to create similar diamonds.

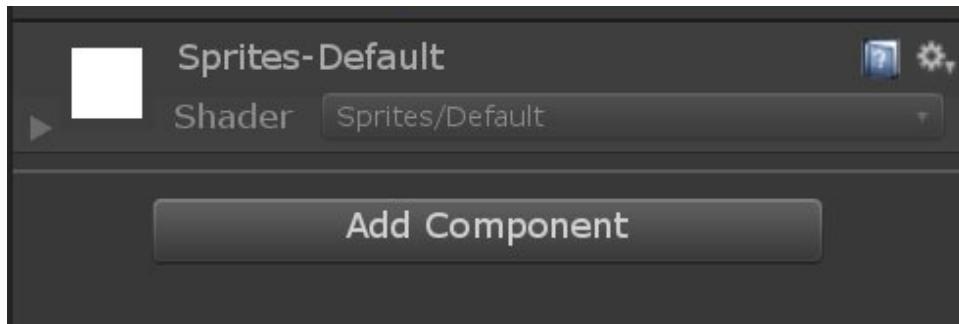




Click on each of the diamonds and look at their script components. You can now set the individual keys in the 'key to destroy' option so that a GameObject destroys itself when that key is pressed. For example, let us create 6 diamonds, and set them to destroy when the A, S, D, F, G, and H keys are pressed.

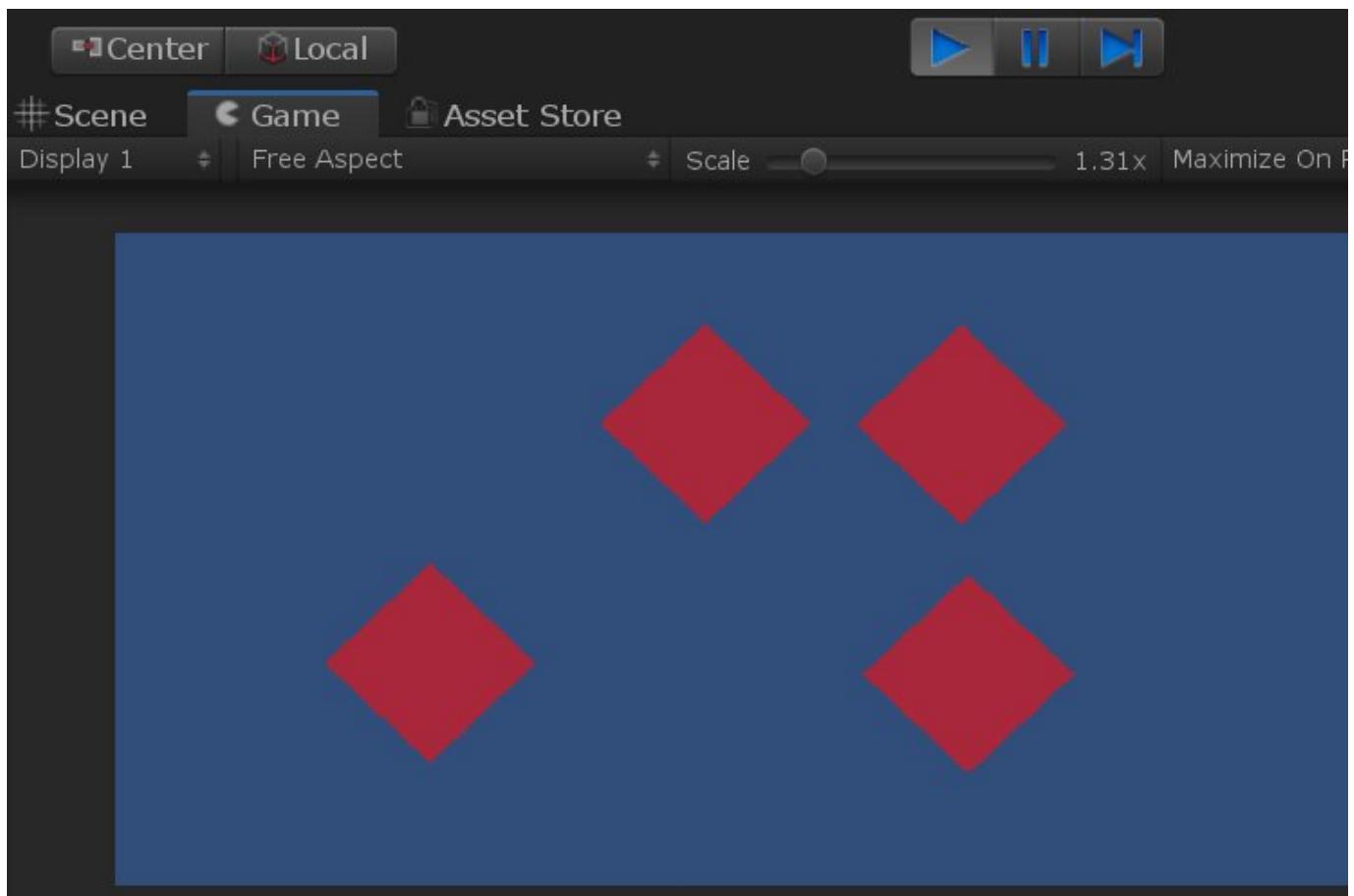






The same key can be set on multiple diamonds, and they will all destroy themselves simultaneously when the key is pressed; this is an example of the use of the `gameObject` reference, which you can use to refer particular objects using the script without having to set them individually.

It is vital to understand that destroying a `GameObject` does not mean an object will explode or shatter. Destroying an object will simply cease its existence as far as the game (and its code) is concerned. The links to this `GameObject` and its references are now broken, and trying to access or use either of them will usually result in errors and crashes.



Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

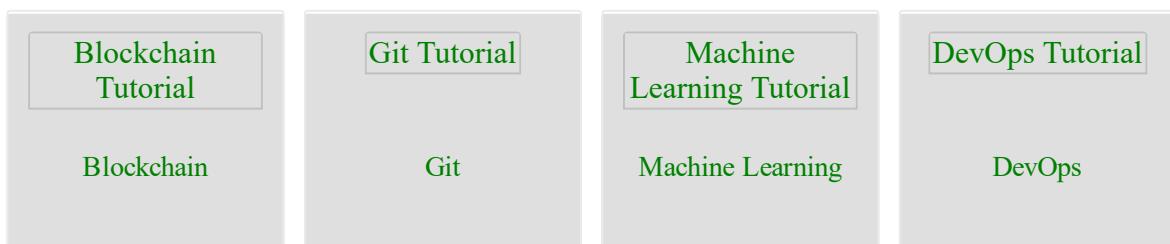
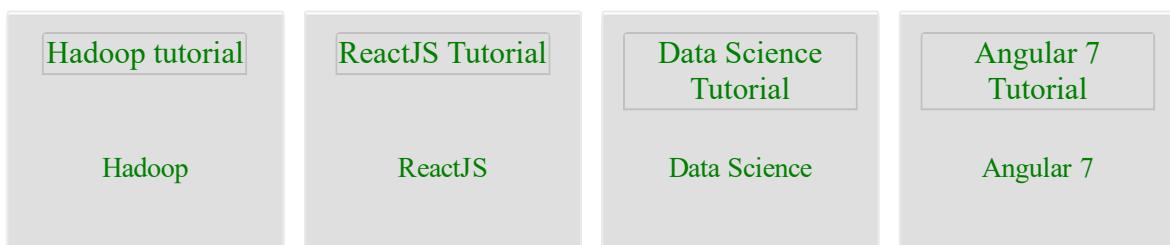
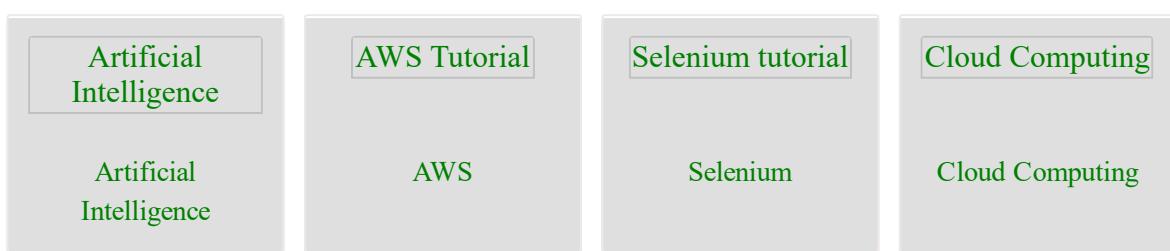
[Keras tutorial](#)

Keras

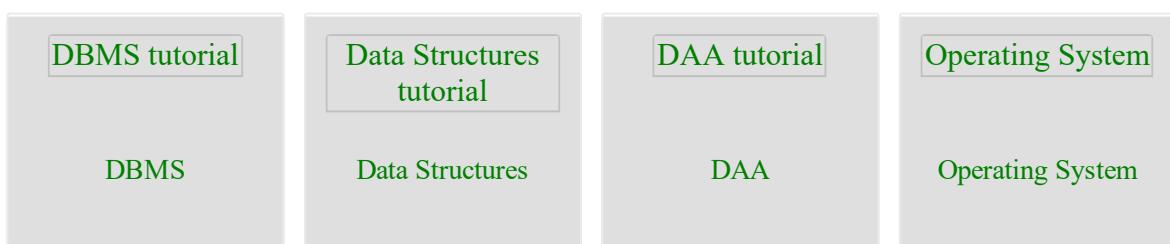
Preparation



Trending Technologies



B.Tech / MCA



Network tutorial
Computer Network

tutorial
Compiler Design

Organization and
Architecture
Computer
Organization

Mathematics
Tutorial
Discrete
Mathematics

Ethical Hacking
Ethical Hacking

Computer
Graphics Tutorial
Computer Graphics

Software
Engineering
Software
Engineering

html tutorial
Web Technology

Cyber Security
tutorial
Cyber Security

Automata
Tutorial
Automata

C Language
tutorial
C Programming

C++ tutorial
C++

Java tutorial
Java

.Net Framework
tutorial
.Net

Python tutorial
Python

List of Programs
Programs

Control Systems
tutorial
Control System

Data Mining
Tutorial
Data Mining

Data Warehouse
Tutorial
Data Warehouse

Unity Console

The console is used to see the output of code. These outputs can be used to quickly test a line of code without having to give added functionality for testing.

Three types of messages usually appear in the default console. These messages can be related to most of the compiler standards:

- o Errors
- o Warnings
- o Messages

Errors: errors are exceptions or issues that will prevent the code from running at all.

Warnings: warnings are also issues, but this will not stop your code from running but may pose issues during runtime.

Messages: messages are outputs that convey something to the user, but they do not usually cause an issue.

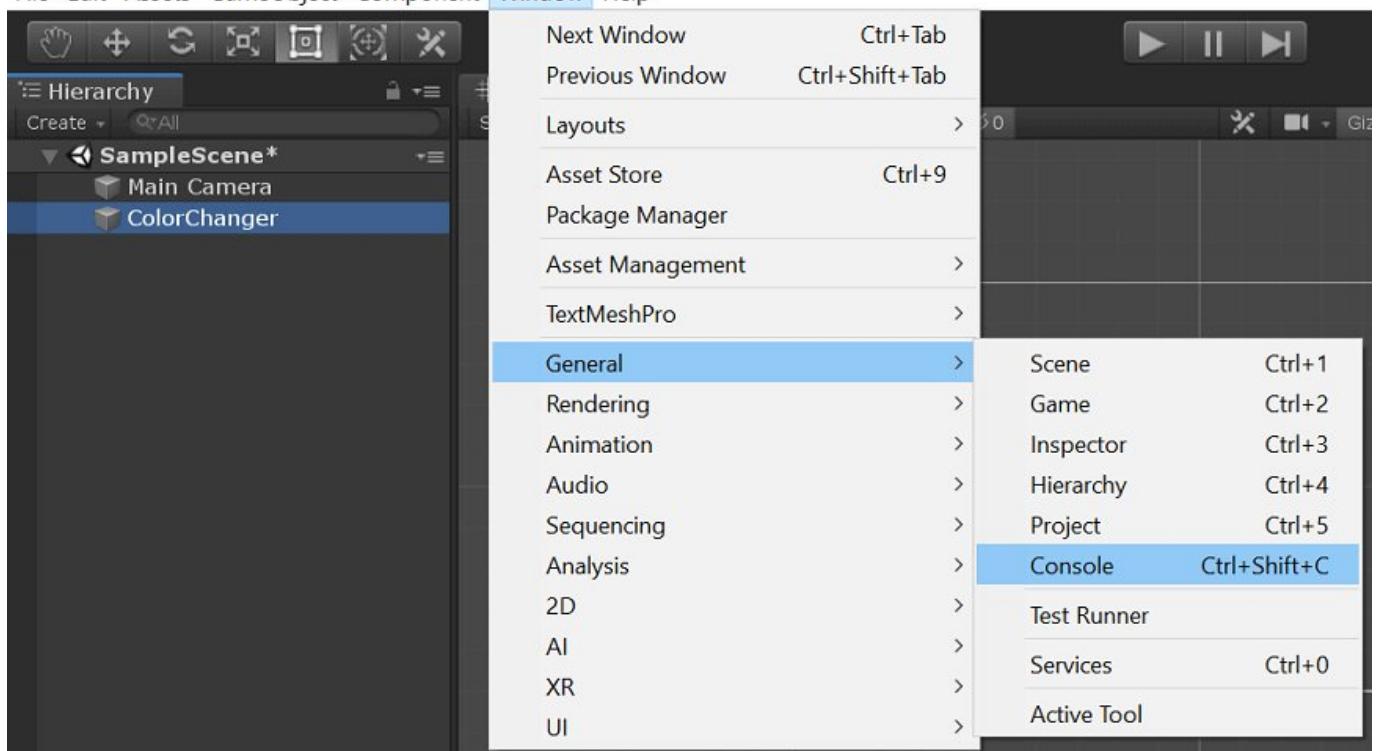
Even we can have the console output our messages, errors, and warnings. To do that, we will use the Debug class.

The Debug class is a part of MonoBehaviour, which gives us methods to write messages to the console, quite similar to how you would create normal output messages in your starter programs.

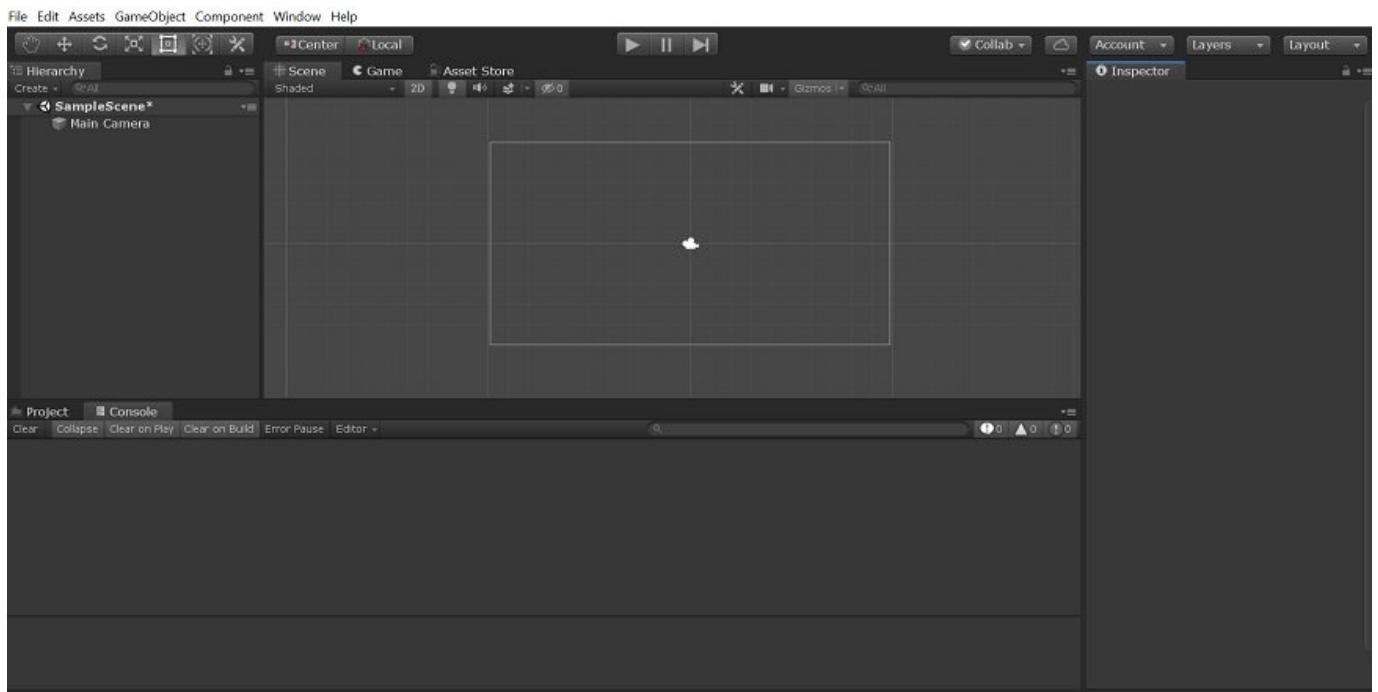
These methods are:

- o Debug.Log
- o Debug.LogWarning
- o Debug.LogError

To open the console from the main menu of Unity Editor, select Windows -> General -> Console or press ctrl + shift + C.



By default console window is at the bottom and next to the Project tab of the Unity editor.



The outputs of the console are more useful to the programmer, not much more useful for the end-user or player.

Let's create a script for displaying a simple message, warning, and Error to the console. These will notify us when the space key, escape key, and delete key was pressed. For this, we will use the Debug class methods, which take in an object as a parameter, which we use a string in.

```

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

public class ConsoleOutput : MonoBehaviour
{
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
            Debug.Log("Message!! Space key was pressed!");

        if (Input.GetKeyDown(KeyCode.Escape))
            Debug.LogWarning("Warning!! Escape key was press

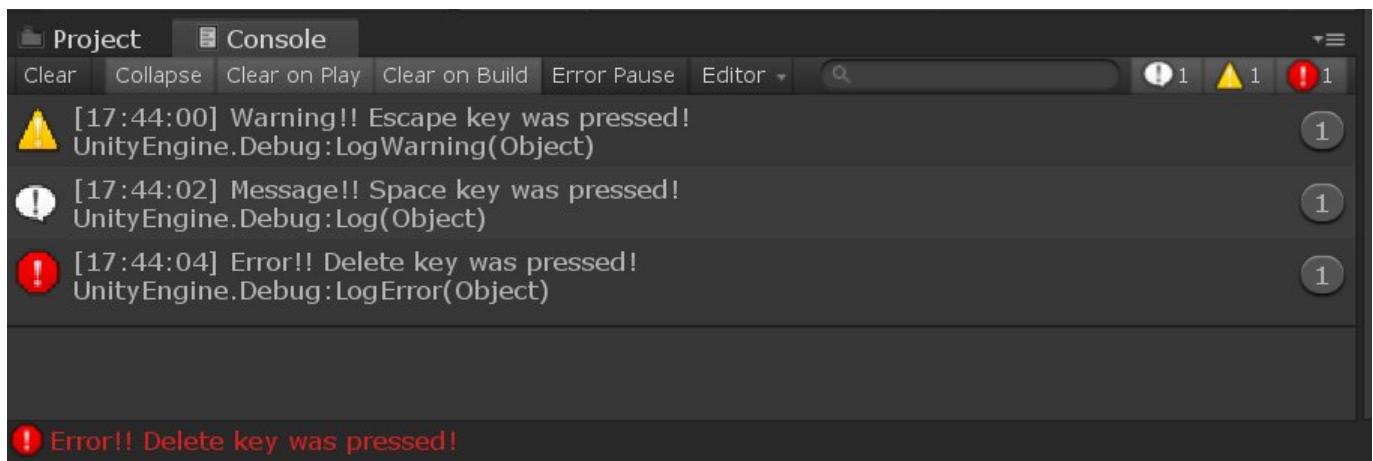
```

```

if (Input.GetKeyDown(KeyCode.Delete))
    Debug.LogError("Error!! Delete key was pressed!");
}
}

```

Output:



← Prev

Next →

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

[Logical](#)

[Verbal Ability](#)

[Interview](#)

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Computer Organization and Architecture

Discrete Mathematics Tutorial

Discrete Mathematics

Computer Network

Compiler Design

Architecture

Computer Organization

Tutorial

Discrete Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse

Unity Sound

Creating the visual elements of a game is only half of the game, adding sounds to your game are just as important as developing amazing shaders. Unity's sound system is flexible and powerful.

Unity can import most standard audio file formats and has features for playing sounds in 3D space, optionally with effects like echo and filtering applied. Even Unity can also record audio from any available microphone on a user's machine for use during gameplay or storage and transmission.

There are two components related to Audio in Unity; they are:

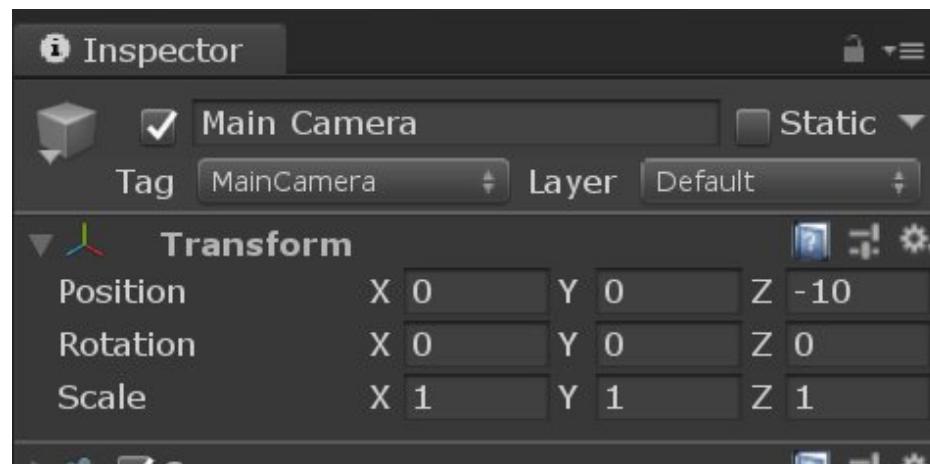
- o Audio Listener
- o Audio Source

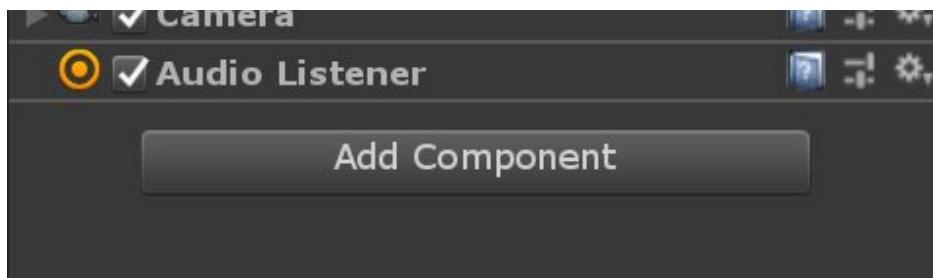
Let's see these components one by one:

Audio Listener

Audio Listener is the component that is automatically attached to the main camera every time you create a scene. It does not have any properties since its only job is to act as the point of perception.

This component listens to all audio playing in the scene and transfers it to the system's speaker. It acts as the ears of the game. Only one AudioListener should be in a scene for it to function properly.

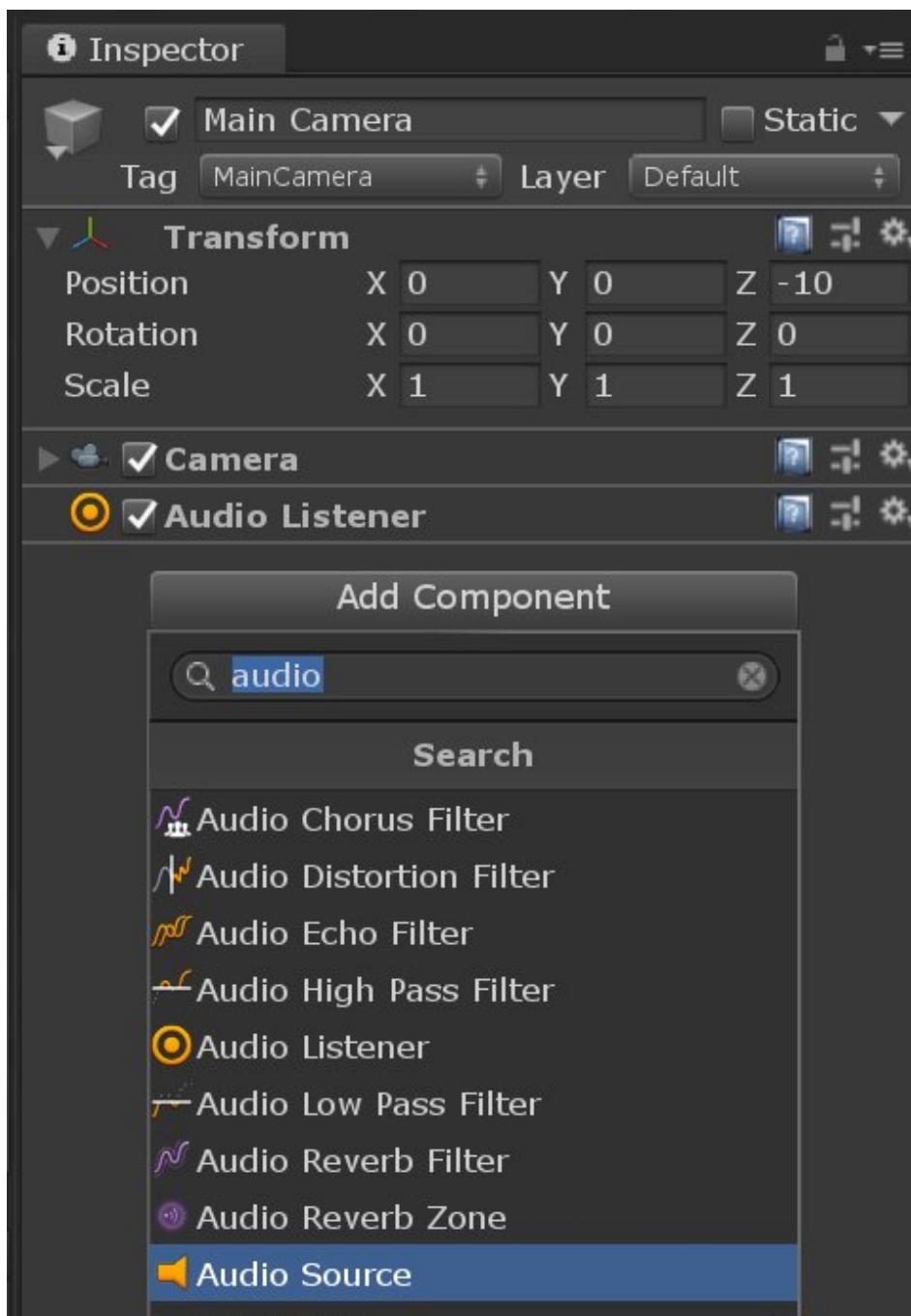




Audio Source

The audio source is the primary component that you will attach to a GameObject to make it play sound. This is the component that is responsible for playing the sound.

To add the Audio Source component, select one GameObject, and go to the Inspector tab. Click on Add Component and search for Audio Source.



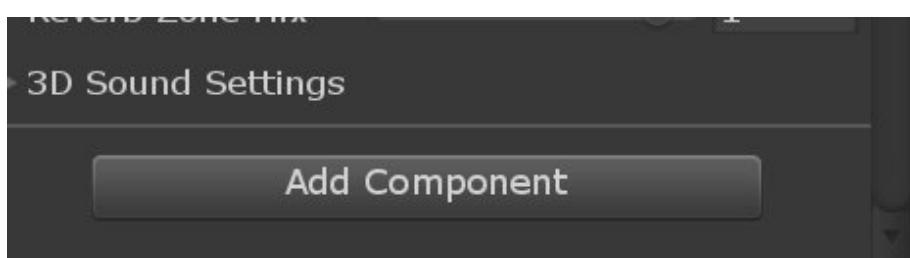
New script

Select Audio Source.

Audio source will playback an Audio Clip when triggered through the mixer, through code or by default, when it awakes.

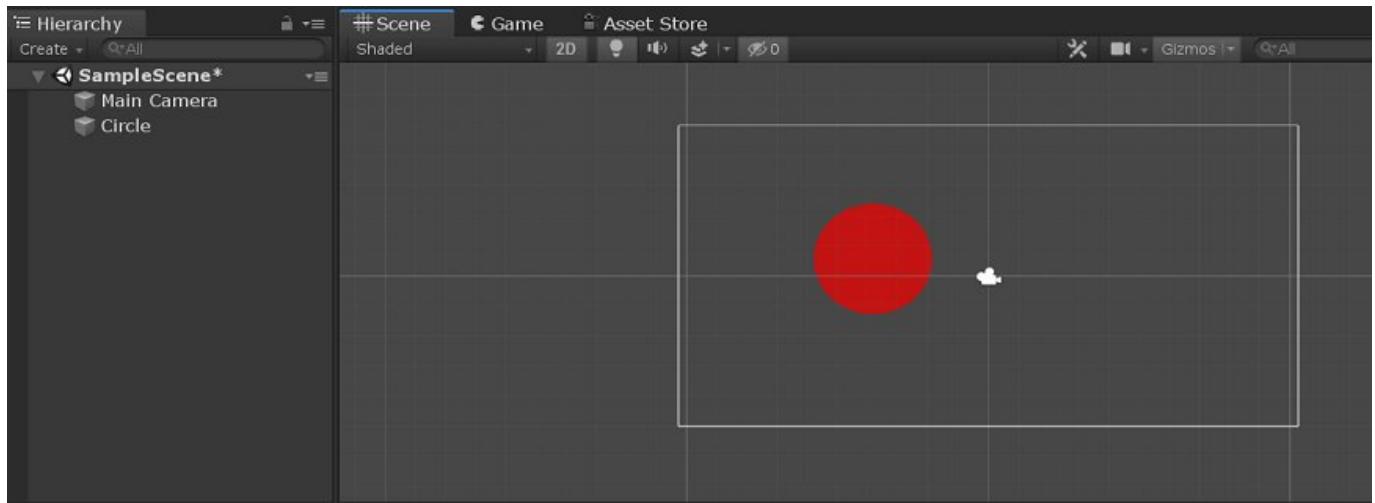
An Audio Clip is a sound file that is loaded into an AudioSource. It can be any standard audio file such as .wav, .mp3, and so on. An Audio Clip is a component within itself.



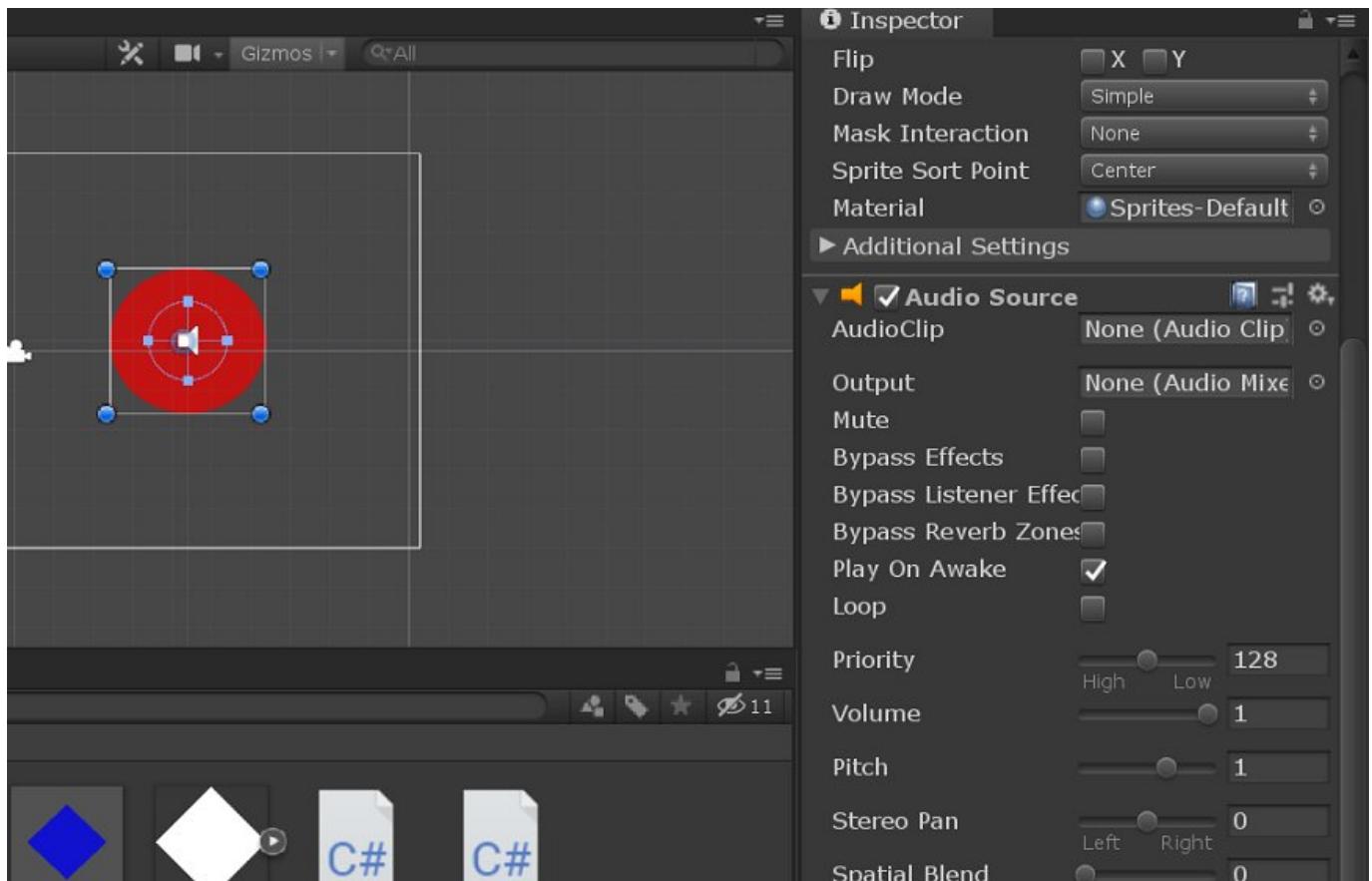


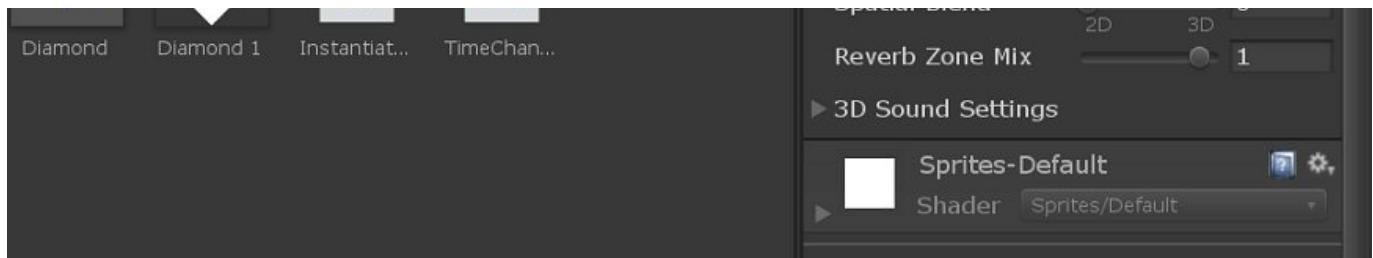
Playing a Sound

Let's add a button that plays a sound when it is clicked. For this, first of all, create a sprite and do one color. Here, I am creating a circle sprite and making it red.



Now, attach an Audio Source component to this circle sprite.

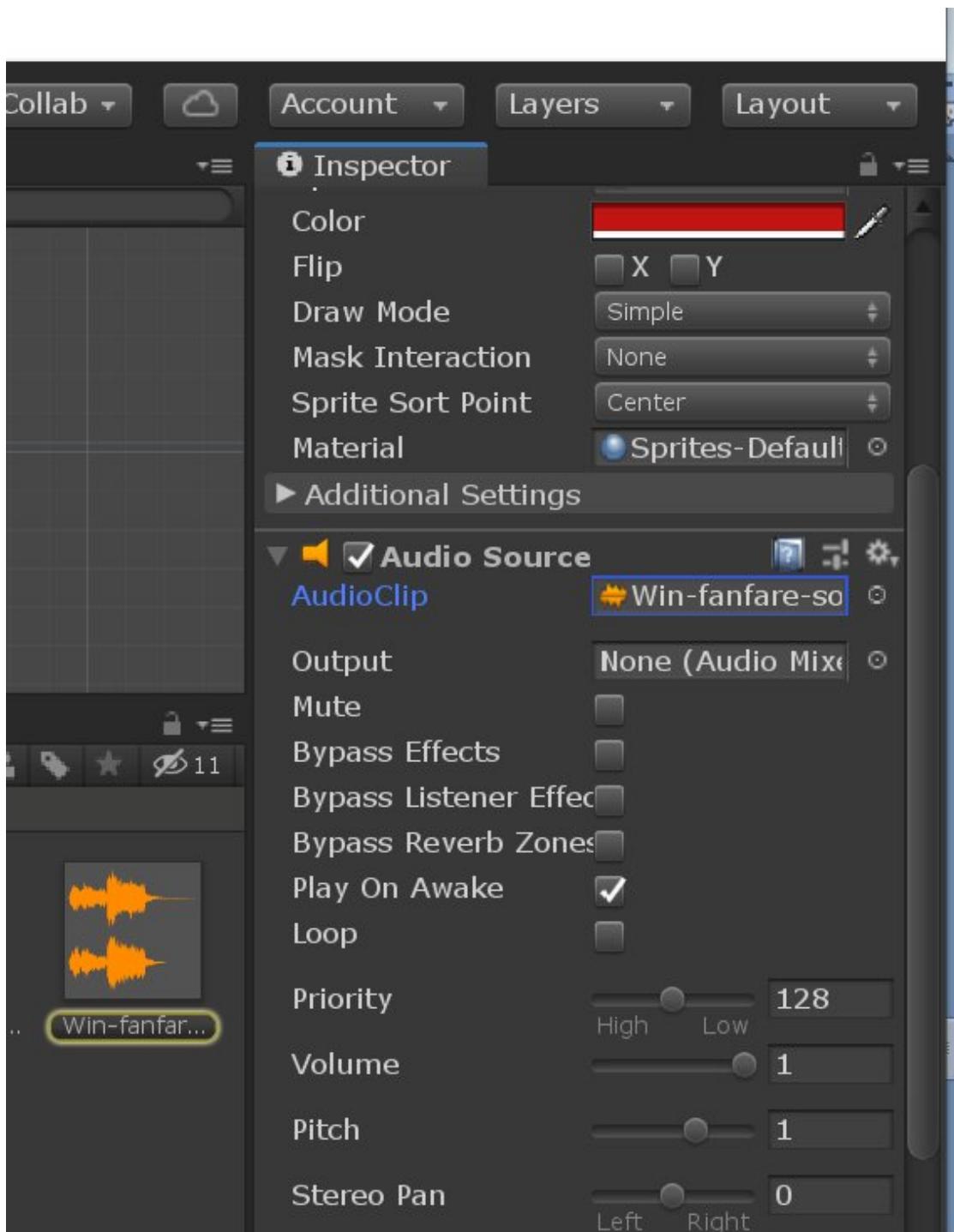


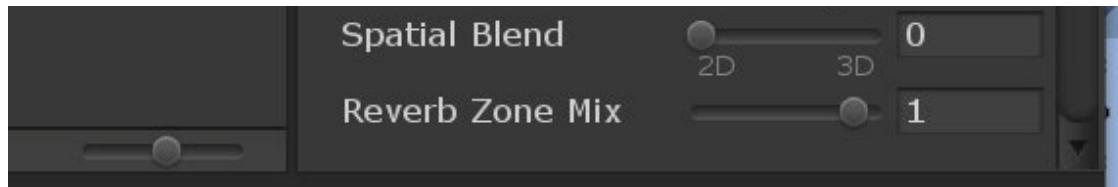


Now, you have to import one audio file. Here, I am downloading success sound. To download it, [click here](#).

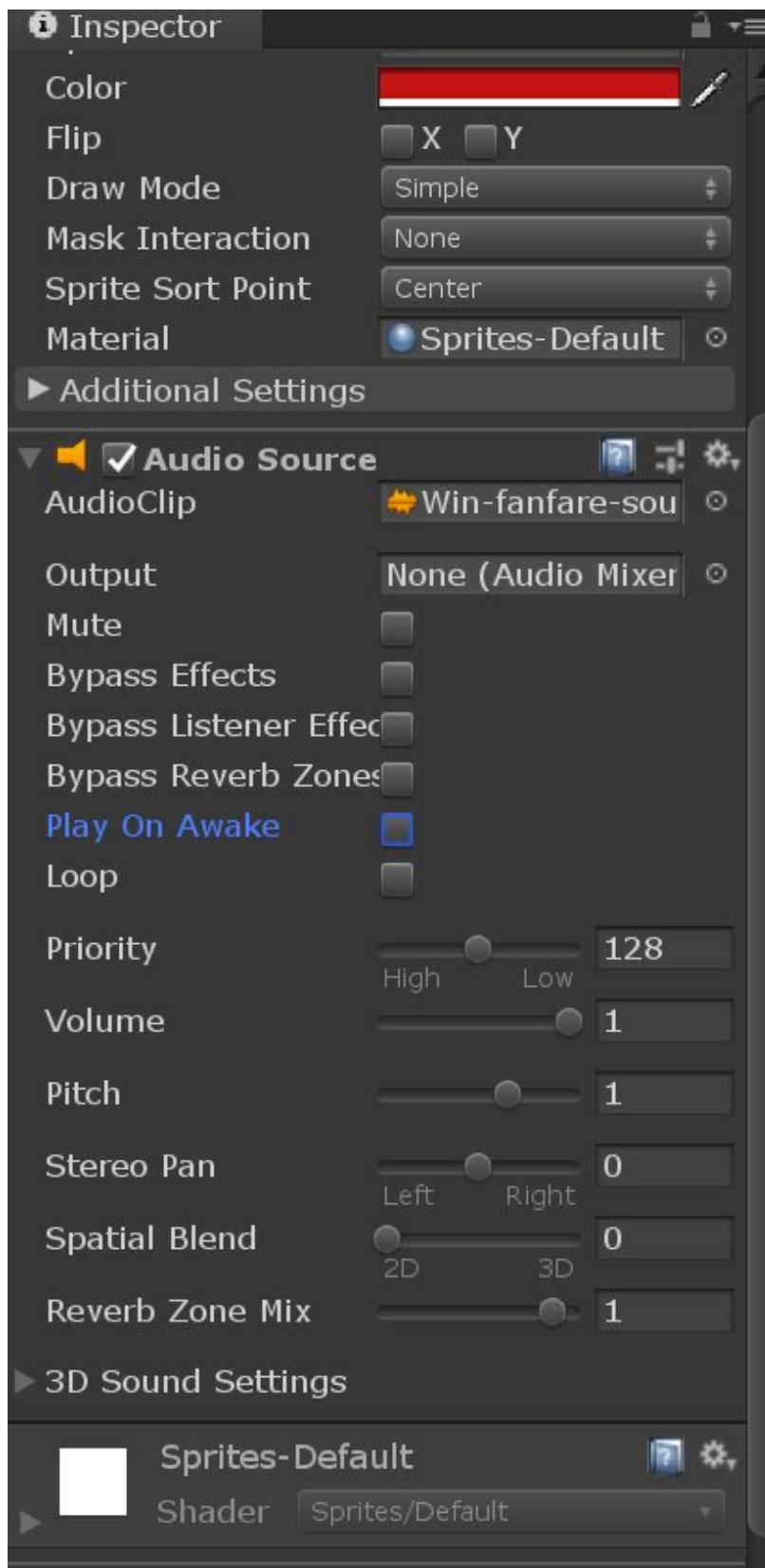
Drag this sound file into Assets.

Drag this sound clip from Assets to the Audio clip slot in our sprite's Audio source component.



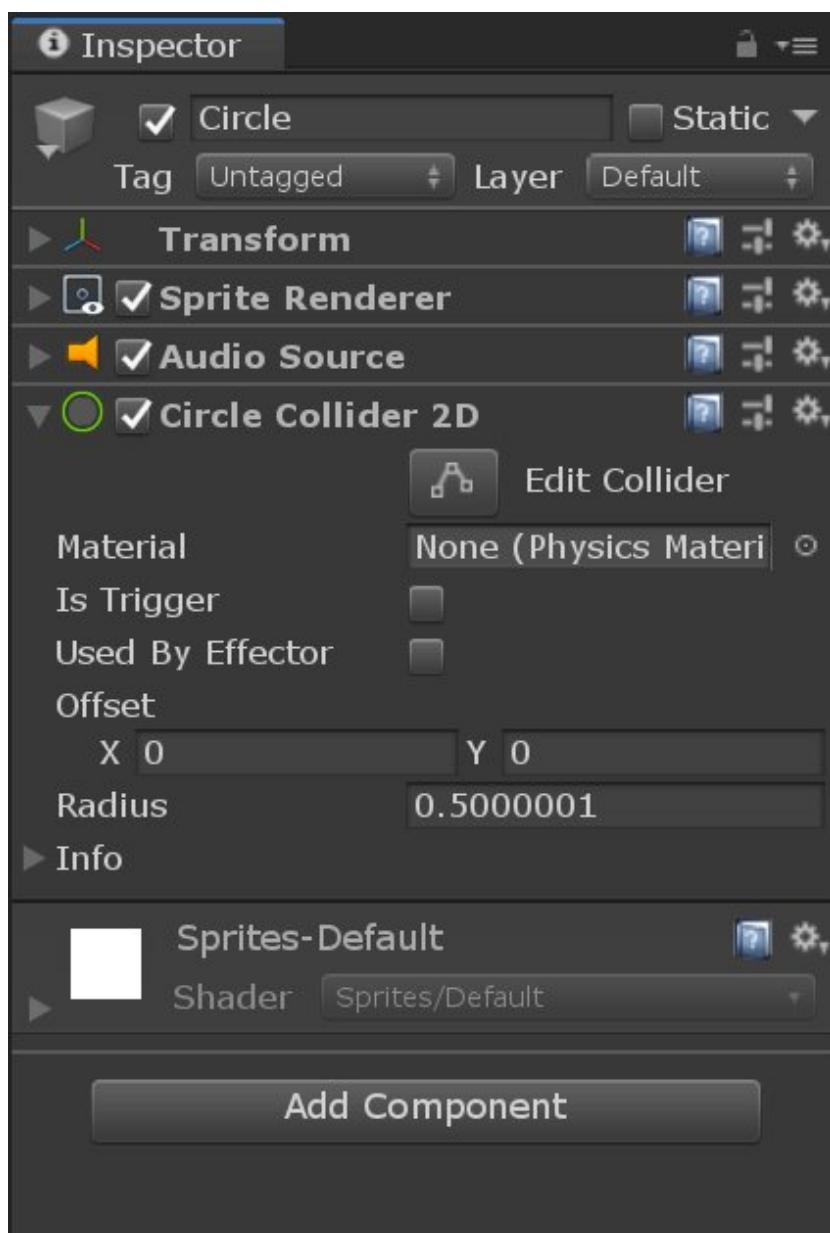


- Uncheck the "Play on Awake" in the audio source properties. Not doing so will make the sound play the moment the game starts.



- Create a new script called "SuccessSound" and open it up.
- In this script file, we have to set up the method to detect the object being clicked. MonoBehaviour gives us just the method we need for it, named onMouseDown. The method is called whenever the mouse clicks in the range of a collider of that gameObject.

For this add 'Circle Collider 2D' component.



- We will not need a Rigidbody for this one; neither do we need to access this collider by code. It just has to be there for the method to work.
- Finally, copy the following code in your script file SuccessSound.cs:

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

using UnityEngine;

```
public class SuccessSound : MonoBehaviour
```

```
{
```

```
    AudioSource mySource;
```

```
    void OnMouseDown() {
```

```
        mySource.Play();
```

```
        Debug.Log("Clicked!");
```

```
}
```

```
// Use this for initialization
```

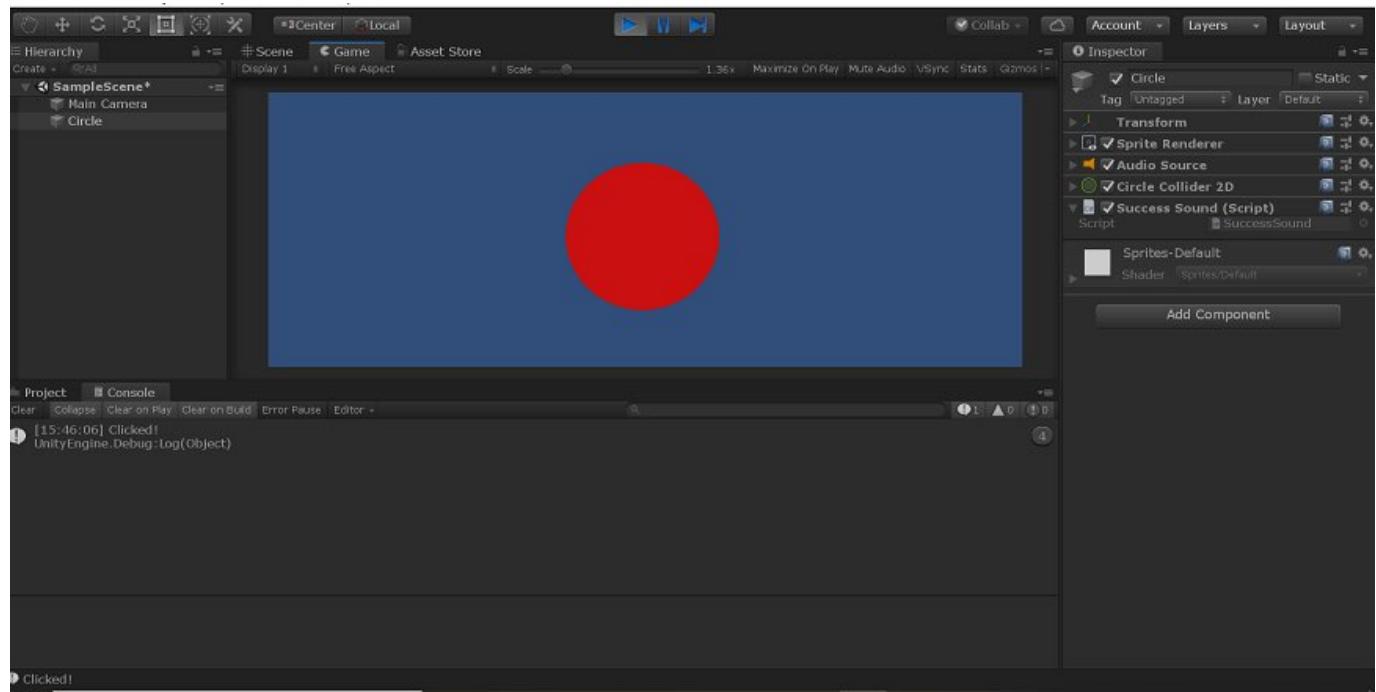
```
    void Start () {
```

```
        mySource = GetComponent<AudioSource>();
```

```
}
```

```
}
```

Attach this script to your circle script. Now play the game. Clicking on the button (red circle sprite) should show a message to the console, and you should hear the success sound.



← Prev

Next →

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials



Preparation

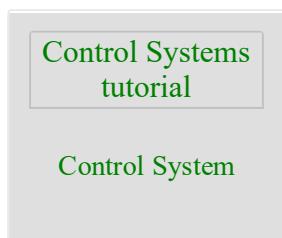
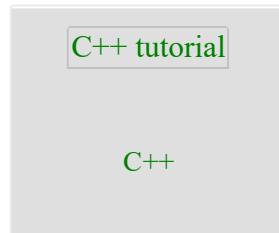
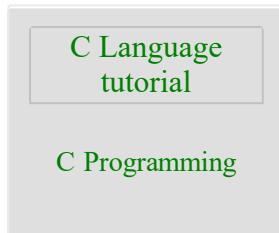
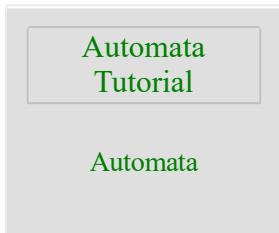
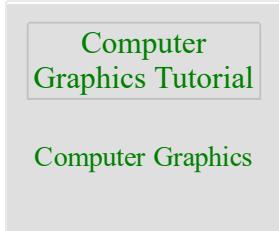
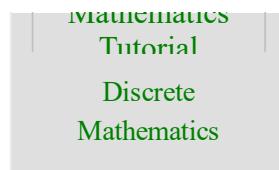
Aptitude	Logical Reasoning	Verbal Ability	Interview Questions
Aptitude	Reasoning	Verbal Ability	Interview Questions
Company Interview Questions			Company Questions

Trending Technologies

Artificial Intelligence	AWS Tutorial	Selenium tutorial	Cloud Computing
Artificial Intelligence	AWS	Selenium	Cloud Computing
Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular 7 Tutorial
Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and	Discrete Mathematics



Unity - Materials and Shaders

Every beautiful looking game contains a different variety of surfaces. Like metal, plastics, holograms, alien artifacts, and so on. Specifically, physical based Rendering.

Rendering in Unity uses Shaders, Materials, and Textures. And three of them have a close relationship.

Materials

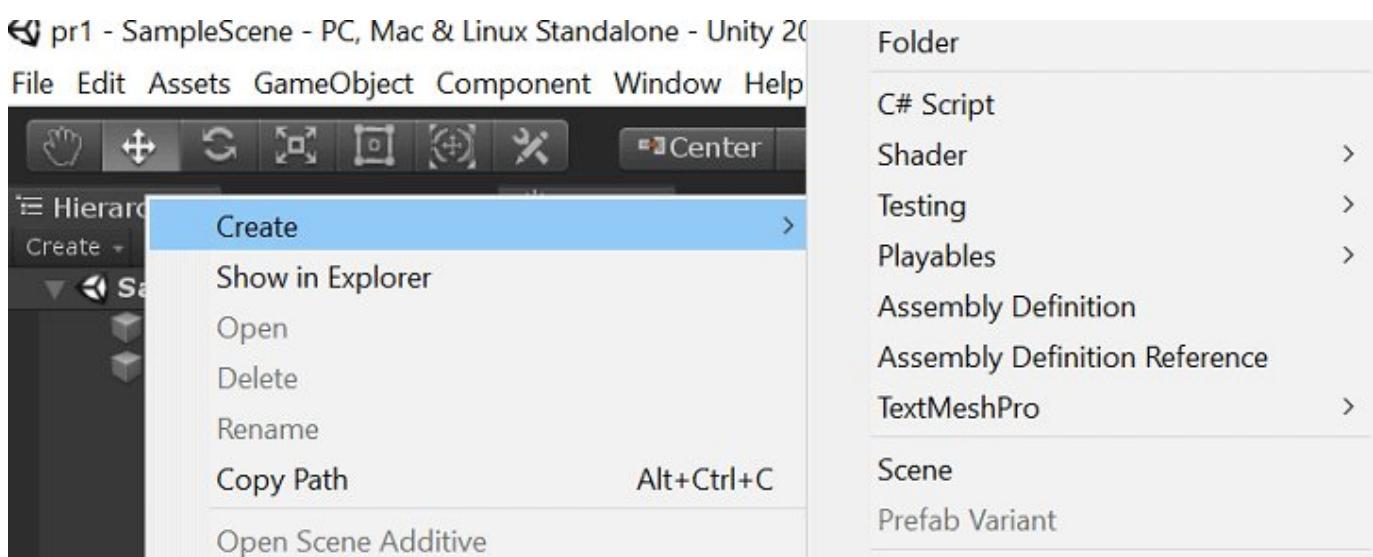
In Unity 3D, a Material is a file that contains information about the lighting of an object with that material.

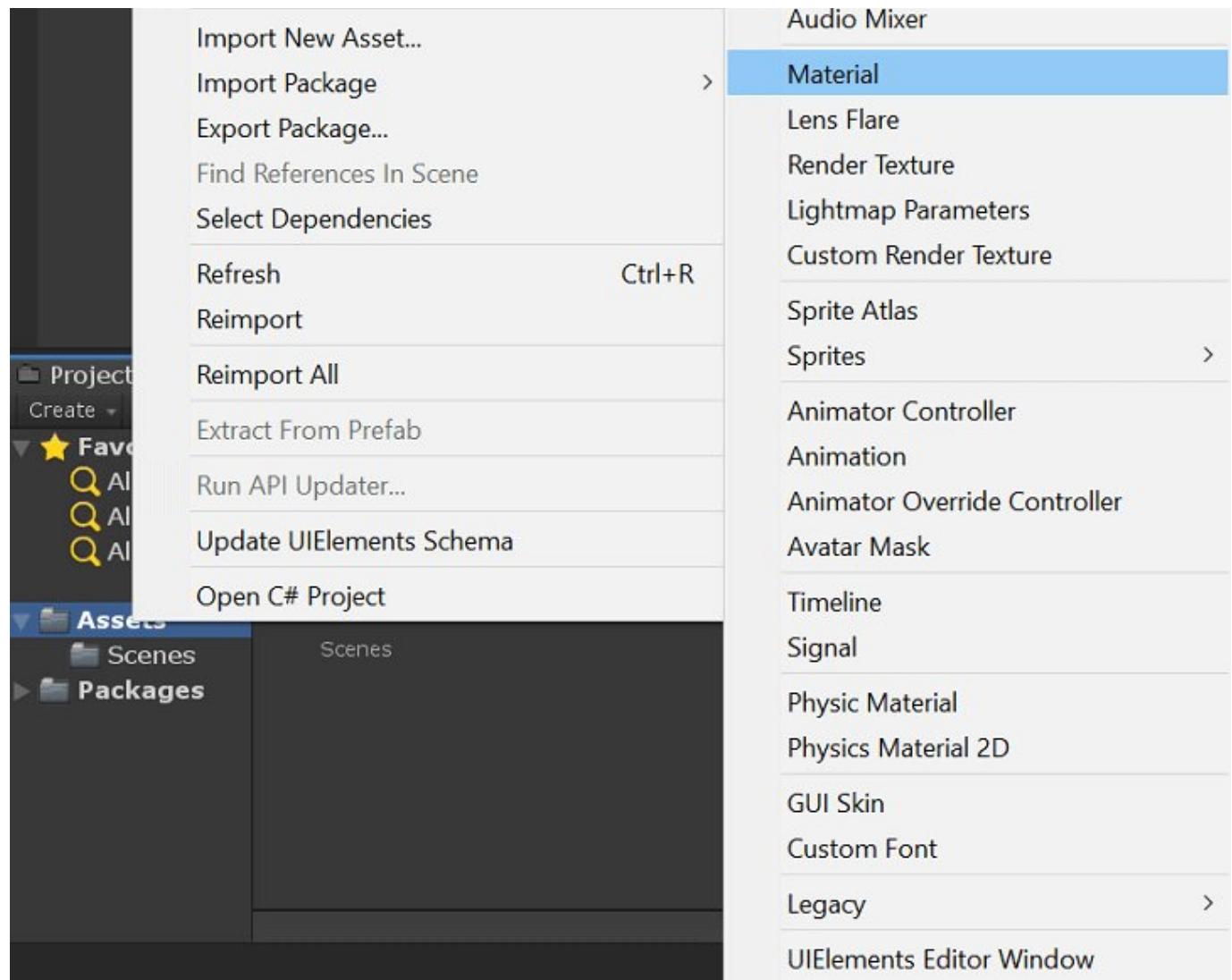
A material has nothing to do with collisions, mass, or even physics in general. It is simply used to define how lighting affects an object with that material.

In unity, Materials are not much more than a container for shaders and textures that can be applied to models. Most of the customization of Materials depends on which shader is selected for it, although all shaders have some common functionality.

Let's create our new material, for that, first of all, create a new 3D project in Unity.

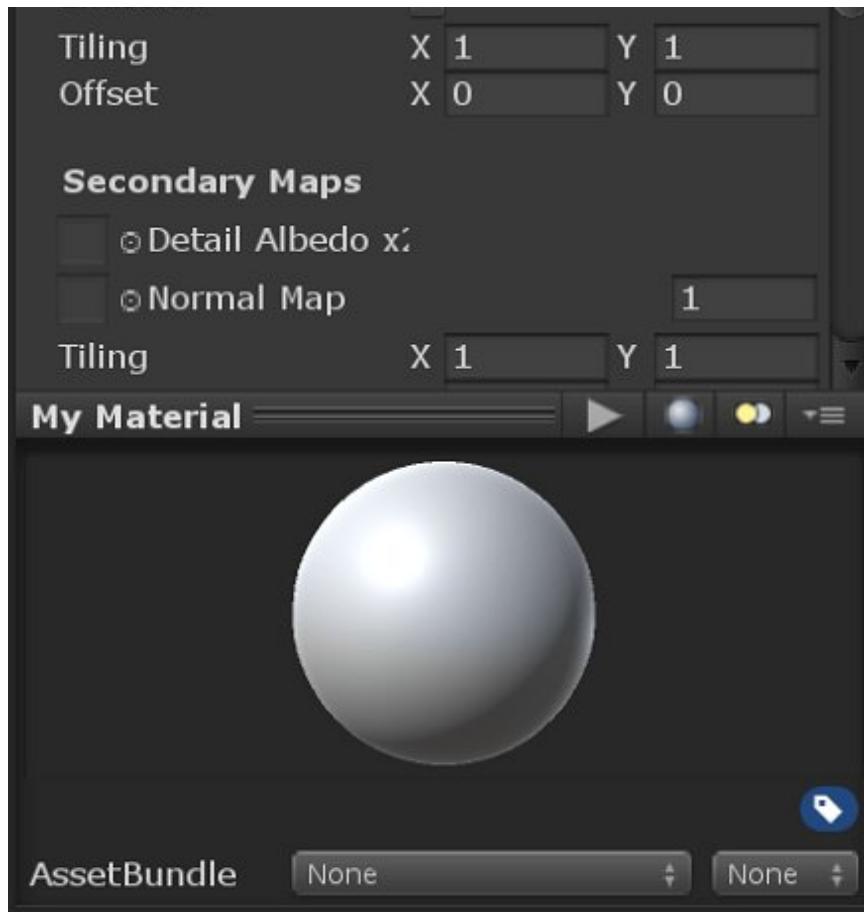
Then right-click on the Assets, and go to Create -> Materials and give a name like 'My Material.'





See the properties of Material.





These properties are not that we have studied so far. That is because these are the properties that are programmed in the shader, not in the material.

Materials are what make our objects visible in the first place. In fact, even in 2D, we use a unique material that does not require lighting as well.

Shaders

A shader is a program that defines how every single pixel is drawn on the screen. Shaders are not programmed in a C# or even in an object oriented programming language at all. Shaders are programmed in a C-like language called GLSL. This language can give direct instructions to the GPU for fast processing.

Shader's scripts have mathematical calculations and algorithms for calculating the color of each pixel rendered, based on the lighting input and the material configuration.

If the texture of a model specifies what is drawn on its surface, the shader is what determines how it is drawn. In other words, we can say that a material contains properties and textures, and shaders dictate what properties and textures a material can have.

Textures

Textures are flat images that can be applied to 3D objects. Textures are responsible for models being colorful and interesting instead of blank and boring.

It looks strange to think that a 2D image can be applied to a 3D model, but it is a very simple and straight forward process once you are familiar with it. Let's see one simple example: think about a water bottle, if you have taken off the label of a water bottle, you would see that it is a flat piece of paper. That label is like a texture. After the label is printed, it is then wrapped around the 3D bottle to provide a more pleasing look.

Just all other assets, adding texture to a Unity is very easy. To create texture in Unity, simply create a folder for your textures; a good name would be Textures. Then download and drag any texture you want in your project into the Textures folder you just created. That's it.

To create a folder, right click on the Assets and go to Create -> Folder. Rename it to 'Textures.' Now download any texture and drag it to that folder.

← Prev

Next →

Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

Tumblr tutorial

Tumblr

React tutorial

ReactJS

Regex tutorial

Regex

Reinforcement learning tutorial

Reinforcement Learning

R Programming tutorial

R Programming

RxJS tutorial

RxJS

React Native tutorial

React Native

Python Design Patterns

Python Design Patterns

Python Pillow tutorial

Python Pillow

Python Turtle tutorial

Python Turtle

Keras tutorial

Keras

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

[Hadoop tutorial](#)

Hadoop

[ReactJS Tutorial](#)

ReactJS

[Data Science Tutorial](#)

Data Science

[Angular 7 Tutorial](#)

Angular 7

[Blockchain Tutorial](#)

Blockchain

[Git Tutorial](#)

Git

[Machine Learning Tutorial](#)

Machine Learning

[DevOps Tutorial](#)

DevOps

B.Tech / MCA

[DBMS tutorial](#)

DBMS

[Data Structures tutorial](#)

Data Structures

[DAA tutorial](#)

DAA

[Operating System](#)

Operating System

[Computer Network tutorial](#)

Computer Network

[Compiler Design tutorial](#)

Compiler Design

[Computer Organization and Architecture](#)

Computer Organization

[Discrete Mathematics Tutorial](#)

Discrete Mathematics

[Ethical Hacking](#)

Ethical Hacking

[Computer Graphics Tutorial](#)

Computer Graphics

[Software Engineering](#)

Software Engineering

[html tutorial](#)

Web Technology

[Cyber Security tutorial](#)

Cyber Security

[Automata Tutorial](#)

Automata

[C Language tutorial](#)

C Programming

[C++ tutorial](#)

C++

[Java tutorial](#)

Java

[.Net Framework tutorial](#)

.Net

[Python tutorial](#)

Python

[List of Programs](#)

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

Data Warehouse
Tutorial

Data Warehouse

Unity Coroutines

A coroutine is a function that allows pausing its execution and resuming from the same point after a condition is met. We can say, a coroutine is a special type of function used in unity to stop the execution until some certain condition is met and continues from where it had left off.

This is the main difference between C# functions and Coroutines functions other than the syntax. A typical function can return any type, whereas coroutines must return an `IEnumerator`, and we must use `yield` before return.

Coroutines can be used for two reasons: asynchronous code and code that needs to compute over several frames.

So basically, coroutines facilitate us to break work into multiple frames, you might have thought we can do this using `Update` function. You are right, but we do not have any control over the `Update` function. Coroutine code can be executed on-demand or at a different frequency (e.g., every 5 seconds instead of every frame).

```
IEnumerator MyCoroutineMethod() {  
    // Your code here...  
  
    yield return null;  
}
```

Example:

```
IEnumerator MyCoroutine()  
{  
    Debug.Log("Hello world");  
    yield return null;  
    //yield must be used before any return
```

```
}
```

Here, the yield is a special return statement. It is what tells Unity to pause the script and continue on the next frame.

We can use yield in different ways:

yield return null - This will Resume execution after All Update functions have been called, on the next frame.

yield return new WaitForSeconds(t) - Resumes execution after (Approximately) t seconds.

yield return new WaitForEndOfFrame() - Resumes execution after all cameras and GUI were rendered.

yield return new WaitForFixedUpdate() - Resumes execution after all FixedUpdatees have been called on all scripts.

yield return new WWW(url) - This will resume execution after the web resource at URL was downloaded or failed.

Start the Coroutine

We can start Coroutine in two ways:

String Based:

Syntax:

```
StartCoroutine( string functionName)
```

Example:

```
StartCoroutine("myCoroutine")
```

IEnumerator Based:

Syntax:

```
StartCoroutine(IEnumerator e)
```

Example:

```
StartCoroutine(myCoroutine())
```

Stop the Coroutine

We can stop the Coroutine using same ways; here instead of calling StartCoroutine, we will use StopCoroutine:

String Based:

Syntax:

```
StopCoroutine( string functionName)
```

Example:

```
StopCoroutine("myCoroutine")
```

IEnumerator Based:

Syntax:

```
StopCoroutine(IEnumerator e)
```

Example:

```
StopCoroutine(myCoroutine())
```

Example 1

Let's see one simple example to see how coroutine works. For that create a circle sprite. Then create one script file, name it as ColorChanger.cs and copy the following code:

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

public class ColorChanger : MonoBehaviour
{
    private SpriteRenderer sr;

    public Color color1;
    public Color color2;

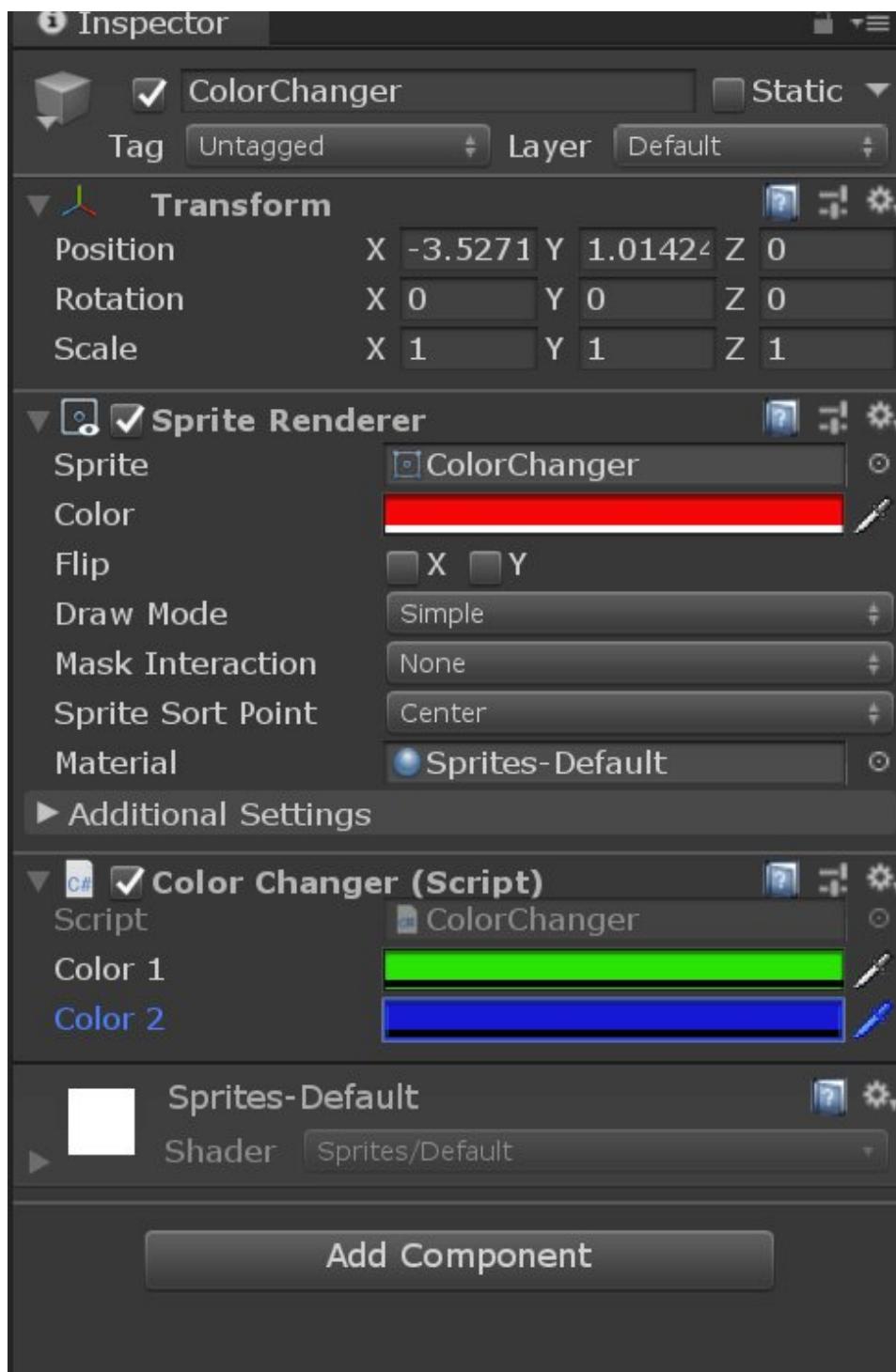
    void Start () {
        sr = GetComponent<SpriteRenderer>();
        StartCoroutine(ChangeColor());
    }

    IEnumerator ChangeColor() {
        while (true) {
            if (sr.color == color1)
                sr.color = color2;

            else
                sr.color = color1;

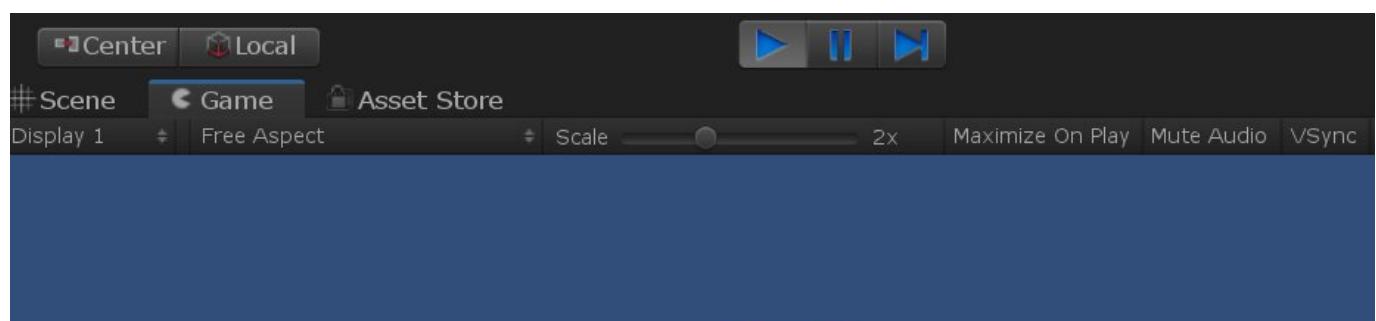
            yield return new WaitForSeconds(3);
        }
    }
}
```

Attach this script file to the Sprite's component. And select two colors in color1 and color2 variable.



Now, when you play this game, our circle object will switch between the two colors in 3 seconds intervals.

Output:





Example 2

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

public class TimeChange : MonoBehaviour
{
    IEnumerator WaitAndPrint()
    {
        // suspend execution for 5 seconds
        yield return new WaitForSeconds(5);
        print("WaitAndPrint " + Time.time);
    }

    IEnumerator Start()
    {
        print("Starting " + Time.time);

        // Start function WaitAndPrint as a coroutine
        yield return StartCoroutine("WaitAndPrint");
        print("Done " + Time.time);
    }
}
```

Output:



```
Project  Console
Clear  Collapse  Clear on Play  Clear on Build  Error Pause  Help

[10:59:18] Starting 0
UnityEngine.MonoBehaviour:print(Object)
[10:59:24] WaitAndPrint 5.033016
UnityEngine.MonoBehaviour:print(Object)
[10:59:24] Done 5.033016
UnityEngine.MonoBehaviour:print(Object)

Done 5.033016
```

← Prev

Next →

[Youtube](#) For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

[React tutorial](#)

[Regex tutorial](#)

[Reinforcement learning tutorial](#)

Tumblr	ReactJS	Regex	Reinforcement Learning
R Programming tutorial	RxJS tutorial	React Native tutorial	Python Design Patterns
R Programming	RxJS	React Native	Python Design Patterns
Python Pillow tutorial	Python Turtle tutorial	Keras tutorial	
Python Pillow	Python Turtle	Keras	

Preparation

Aptitude	Logical Reasoning	Verbal Ability	Interview Questions
Aptitude	Reasoning	Verbal Ability	Interview Questions
Company Interview Questions			Company Questions

Trending Technologies

Artificial Intelligence	AWS Tutorial	Selenium tutorial	Cloud Computing
Artificial Intelligence	AWS	Selenium	Cloud Computing
Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular 7 Tutorial
Hadoop	ReactJS	Data Science	Angular 7

Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial
Cyber Security	Automata	C Programming	C++
Java tutorial	.Net Framework tutorial	Python tutorial	List of Programs
Java	.Net	Python	Programs
Control Systems tutorial	Data Mining Tutorial	Data Warehouse Tutorial	
Control System	Data Mining	Data Warehouse	



Game programming

Lecture 7,8: **Unity UI**

Marwa Al-Hadi



اول لعبه يمنية تحتري الشارع اليمني

Unity UI

Unity UI (User Interface) is used to create a user interface in your game or application.

The workflow for designing Unity UI follows a slightly different path than the one we have been going through so far. For starters, UI elements are not standard GameObjects and can't be used as such. UI elements are designed differently; UI elements are designed differently; a menu button that looks correct in a 4:3 resolution may look stretched or distorted in a 16:9 resolution if not set upright.

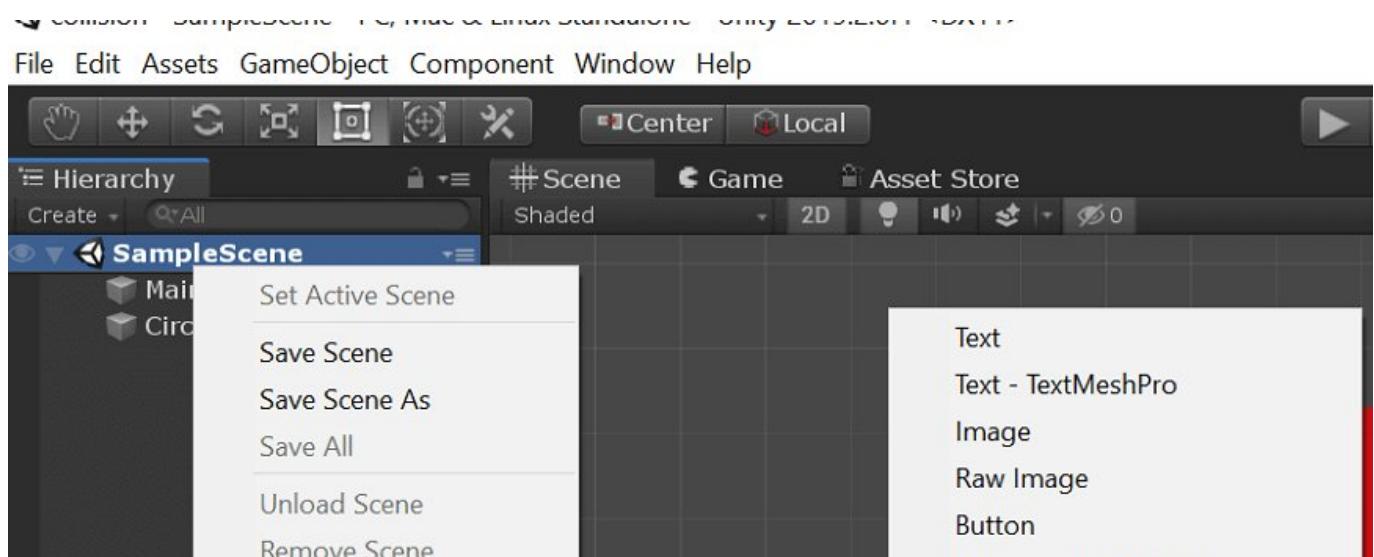
In Unity, we cannot place the UI elements directly on the scene. They are always located as children of a special GameObject called the **Canvas**.

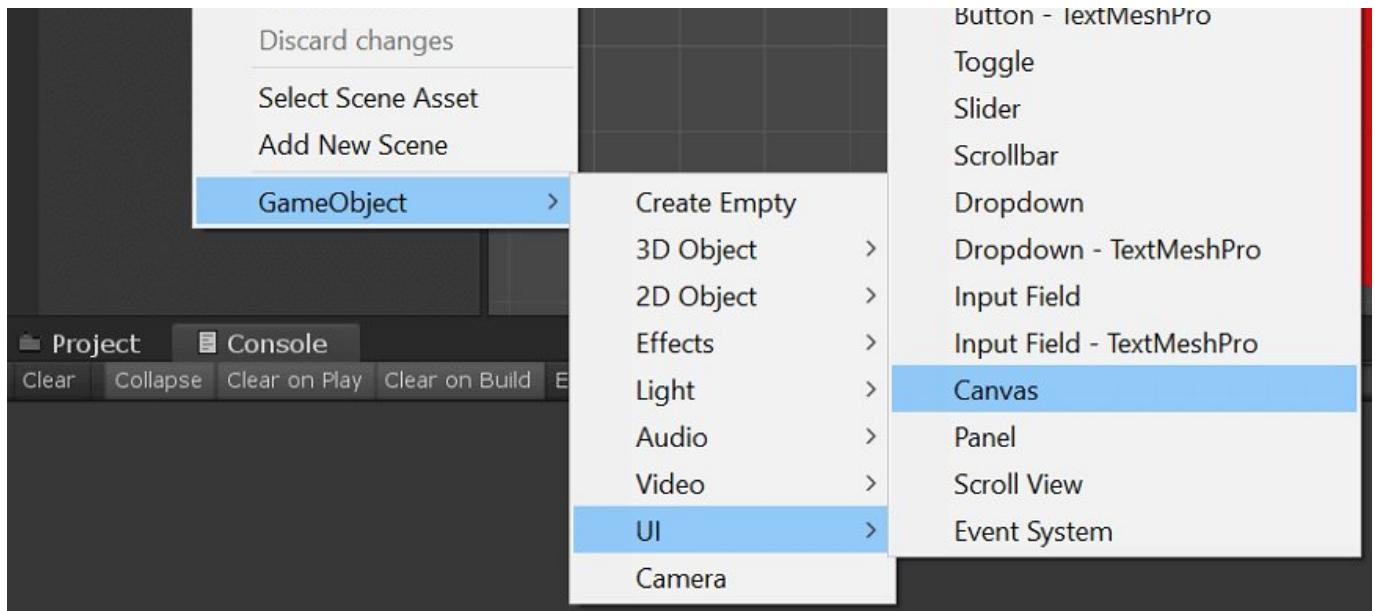
UI Canvas

The UI Canvas is like a drawing sheet for UI elements on the scene, where all UI elements will render. When you create a UI element without an existing canvas will automatically generate one.

UI Canvas acts as the master of all UI elements on the screen. Because of that, all UI elements are required to be the child game object of the canvas game object.

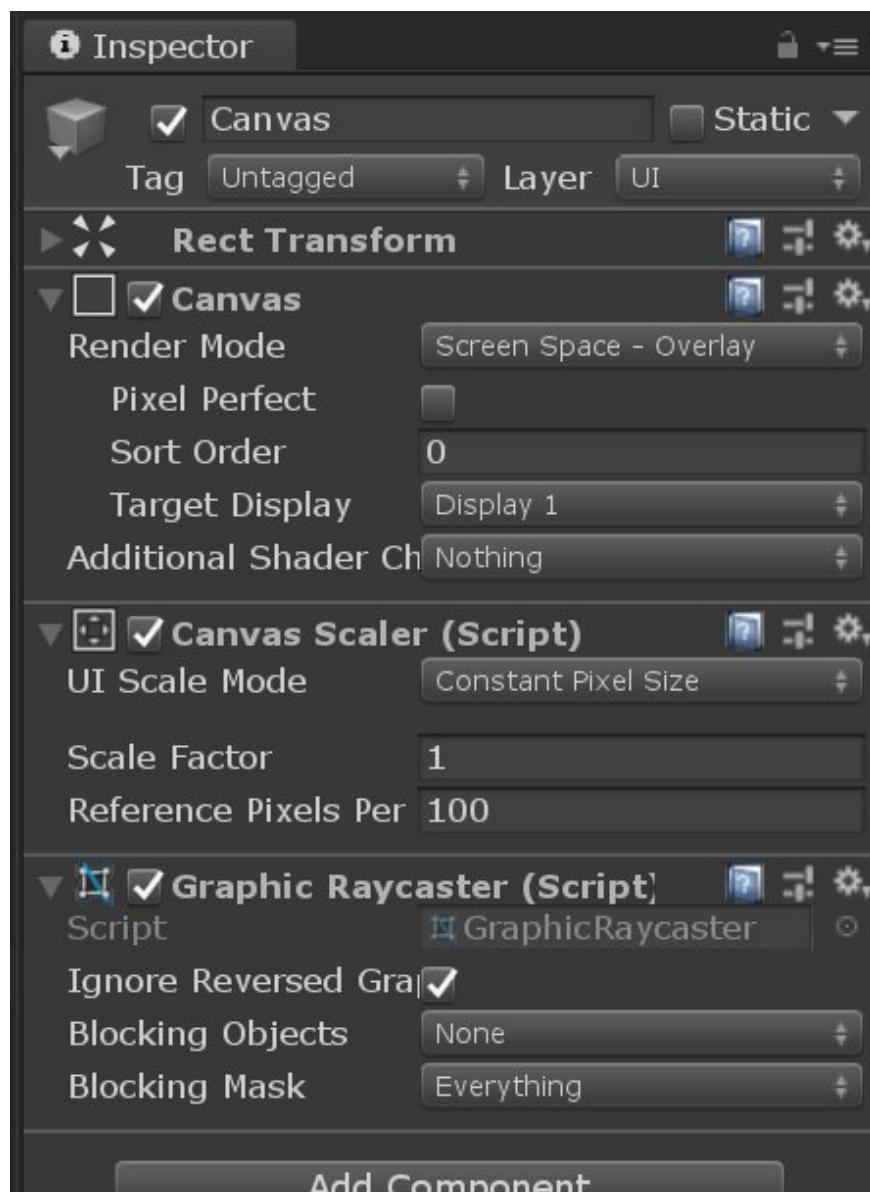
To add the canvas in your scene right, click on the Scene name or Main Camera from the Hierarchy tab and select GameObject -> UI -> Canvas.

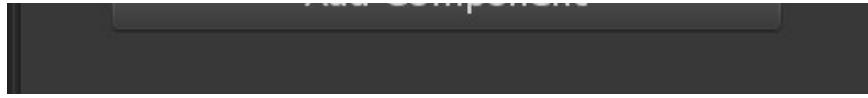




Canvas Components

Let's see the canvas components:

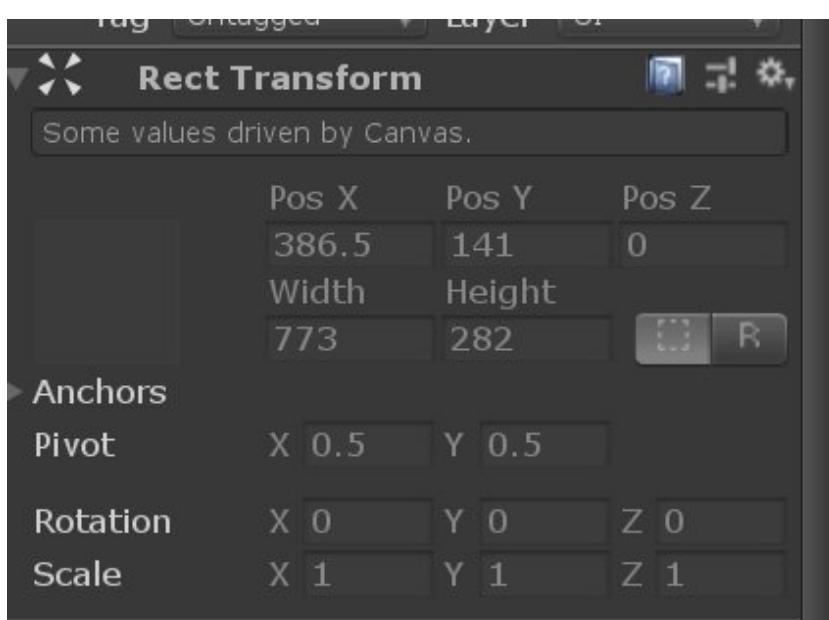




RectTransform

This is the **top component of the canvas gameObject**. It has many new **properties** that a standard GameObject's transform does not have.

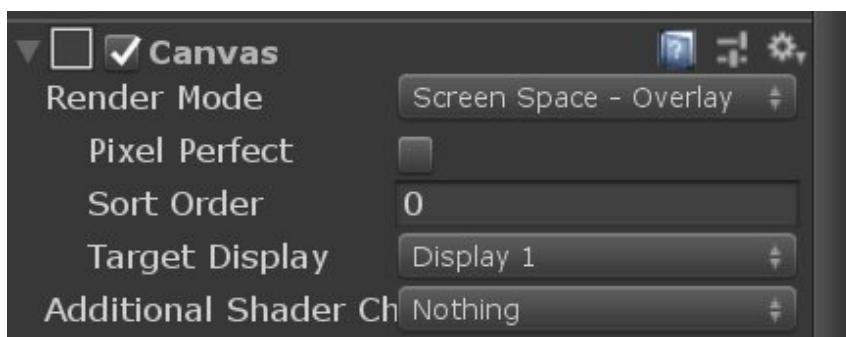
This is because while a normal GameObject's Transform describes an **imaginary point in 3D space**, a RectTransform defines an **imaginary rectangle**. This means we need other properties for defining exactly where the rectangle is, how big it is, and how it is oriented.



Canvas Component

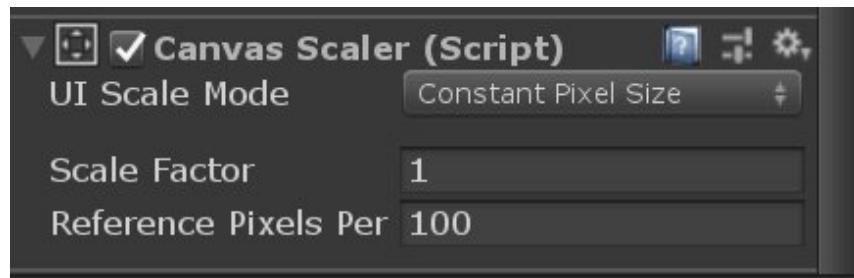
Canvas is the **master component** that holds a couple of universal options as to how the UI is drawn.

The **first property** **Render Mode** defines the method that is **used to draw the canvas onto the Game's view**.



Canvas Scalar

The canvas scalar component is a set of options that allows you to [adjust the scale and appearance of the UI elements in a more definitive way](#).



Graphics Raycaster

The Graphics Raycaster component deals primarily with raycasting [\(link to Unity Documentation for Raycasting\)](#) the UI elements and ensuring [user-initiated events](#) like clicks and drags work correctly.



← Prev

Next →

[Youtube](#) For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share





Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

Aptitude

[Logical Reasoning](#)

Reasoning

[Verbal Ability](#)

Verbal Ability

[Interview Questions](#)

Interview Questions

[Company Interview Questions](#)

Company Questions

Trending Technologies

Artificial Intelligence	AWS Tutorial	Selenium tutorial	Cloud Computing
Artificial Intelligence	AWS	Selenium	Cloud Computing
Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular 7 Tutorial
Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial

Cyber Security

Automata

C Programming

C++

Java tutorial

Java

.Net Framework
tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

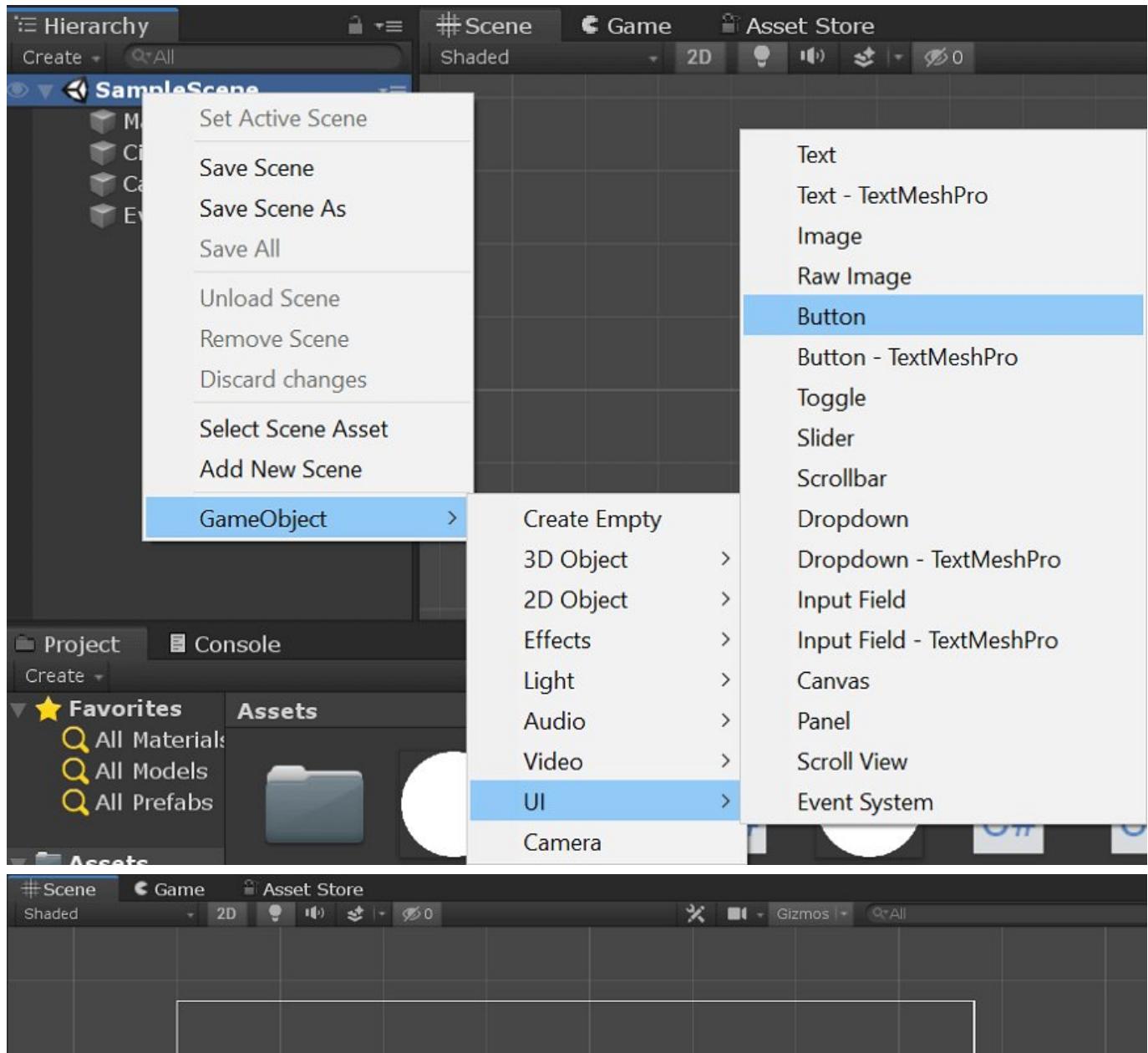
Data Warehouse
Tutorial

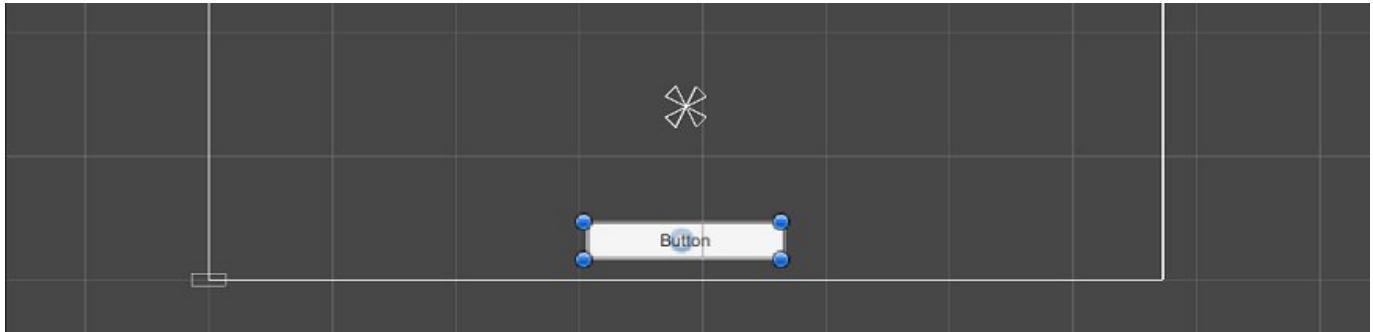
Data Warehouse

Unity UI Button

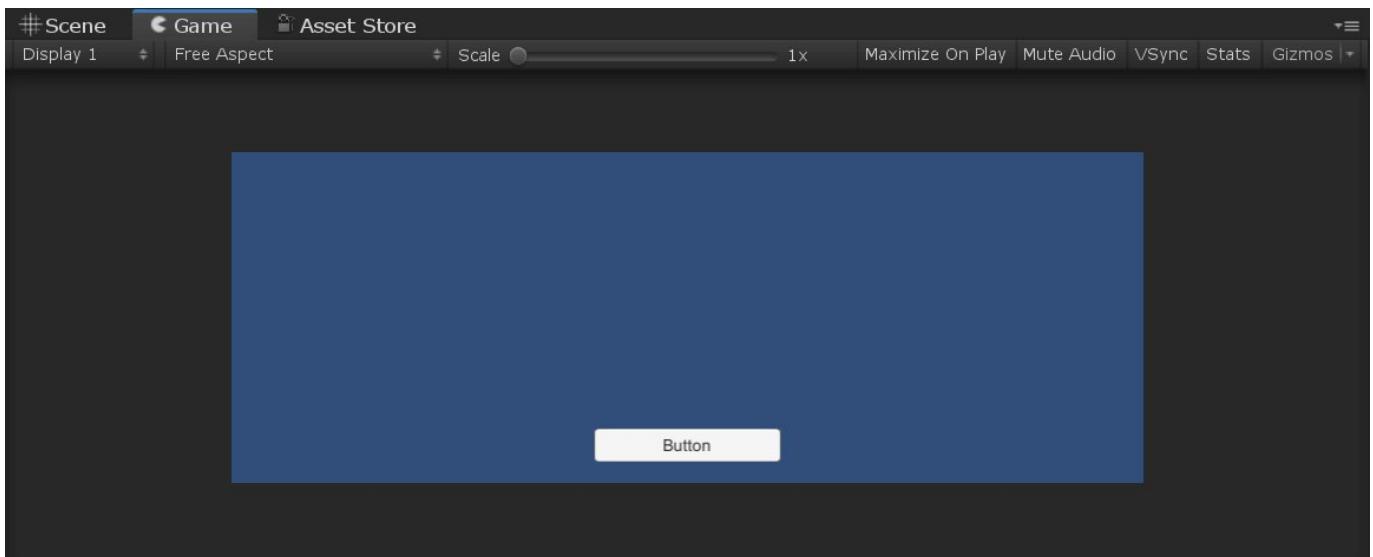
The button UI element in unity **responds to a click from the user** and is used to initiate or confirm **an action**. Some common examples are: **Submit, Ok, and Cancel buttons**.

To insert a button element, right in the scene hierarchy, and select **GameObject -> UI -> Button**. If you don't have an existing canvas and an EventSystem, Unity will automatically create one for you, and place the button inside the canvas as well.





Game tab:



When you play the scene, you will notice that the button already **has some standard functionality**, such as detecting when the mouse is hovering over it and changing color when you pressed that button.

First of all, create a script file, and rename it to `ButtonActions.cs`

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

public class ButtonAction : MonoBehaviour
{
    int n;

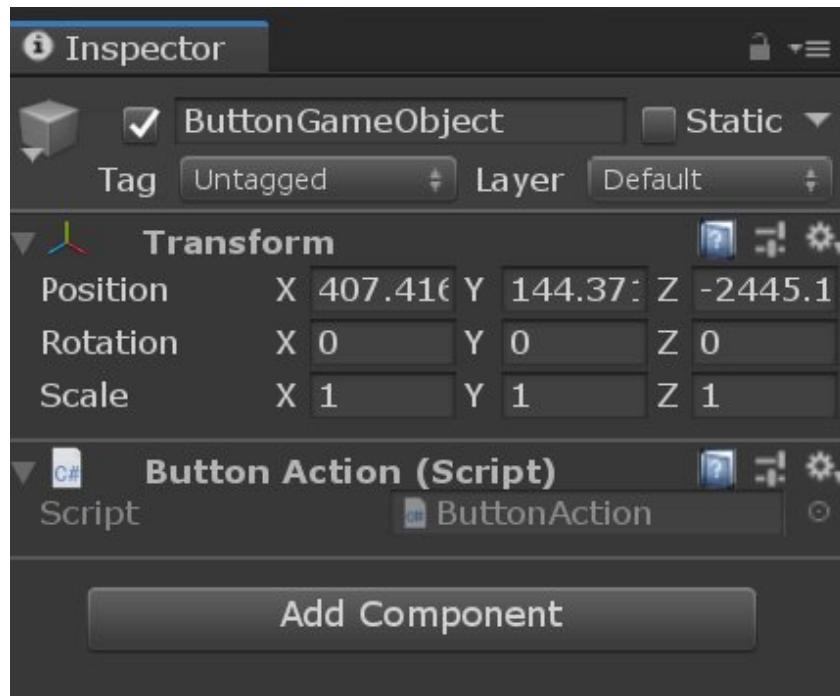
    public void OnButtonPress(){
        n++;
        Debug.Log("Button clicked " + n + " times.");
    }
}
```



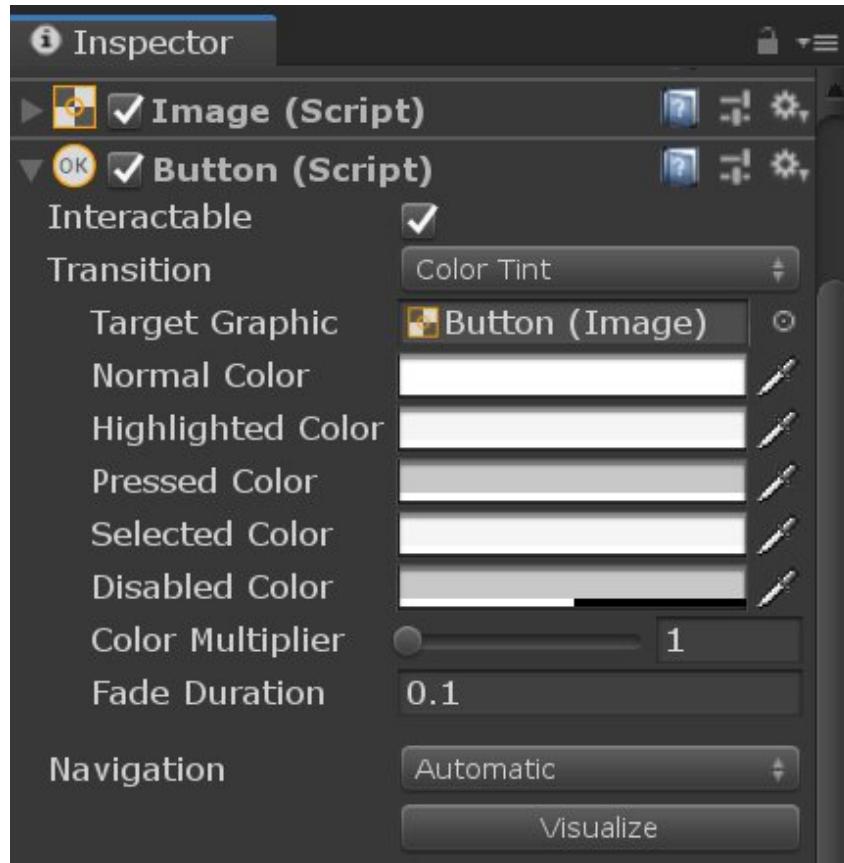
```
}
```

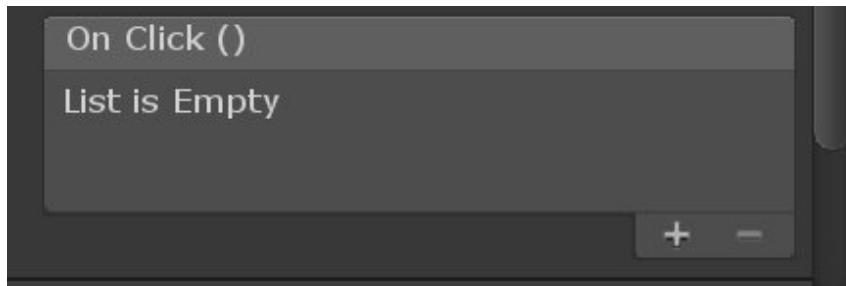
This script specified a simple method that logs how many times we have hit the button.

Now, create an empty GameObject and attach this script file to it. We do this because a button will not do anything on its own; it only calls the specified method in its scripting.

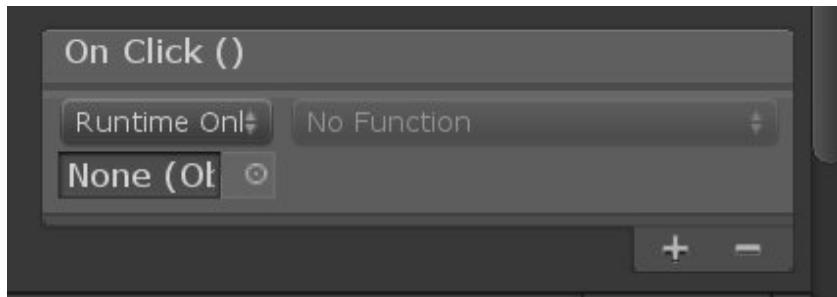


Now, select the Button, and go to the Inspector tab and search for `OnClick()` property.





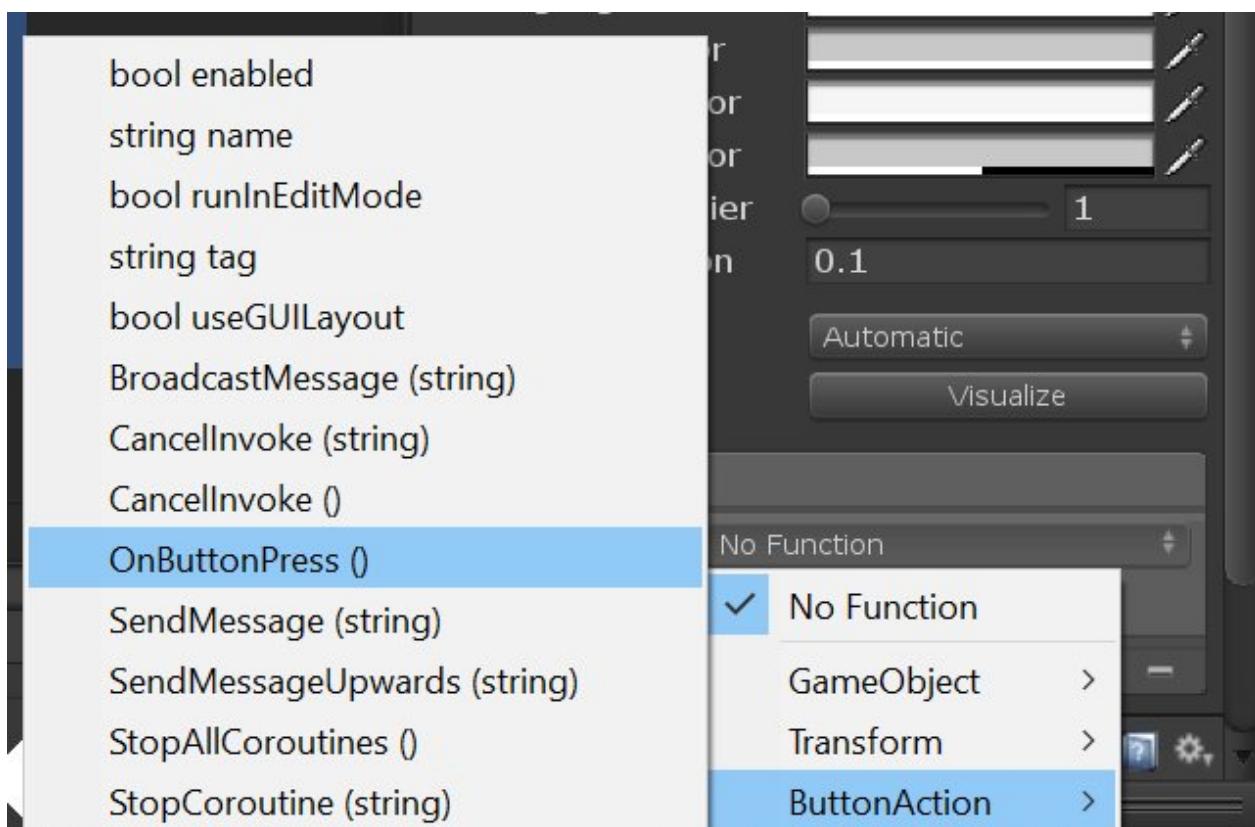
Click on that + icon on the bottom tab, and when you click, a new entry should show up in the list.



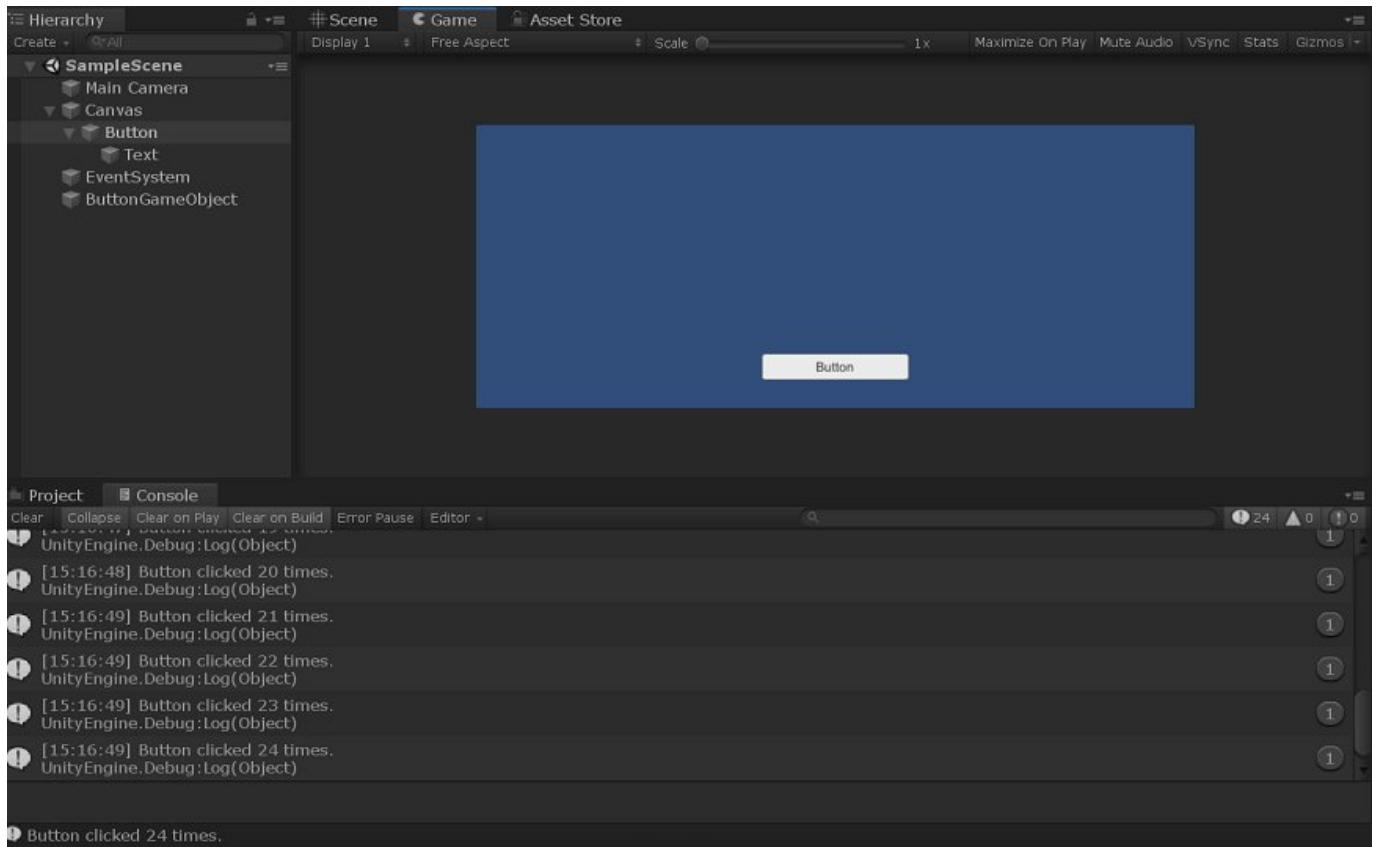
This entry defines what object the button press acts on, and what function of that object's script is called. Because of the event system used in the button press, you can generate multiple functions simply by adding them to the list.

Drag and drop the empty GameObject, which contains the ButtonGameObject script we created, onto the None (Object) slot.

Navigate the 'No Function' dropdown list, and look for our OnButtonPress method. For that, go to your Script file name (ButtonAction) and select OnButtonPress () method.



When you play the game now, you can test the button, the console prints out how many times you have pressed the button.



← Prev

Next →

[Youtube](#) For Videos Join Our Youtube Channel: [Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation

[Aptitude](#)

Aptitude

[Logical Reasoning](#)

Reasoning

[Verbal Ability](#)

Verbal Ability

[Interview Questions](#)

Interview Questions

[Company Interview Questions](#)

Company Questions

Trending Technologies

Artificial Intelligence	AWS Tutorial	Selenium tutorial	Cloud Computing
Artificial Intelligence	AWS	Selenium	Cloud Computing
Hadoop tutorial	ReactJS Tutorial	Data Science Tutorial	Angular 7 Tutorial
Hadoop	ReactJS	Data Science	Angular 7
Blockchain Tutorial	Git Tutorial	Machine Learning Tutorial	DevOps Tutorial
Blockchain	Git	Machine Learning	DevOps

B.Tech / MCA

DBMS tutorial	Data Structures tutorial	DAA tutorial	Operating System
DBMS	Data Structures	DAA	Operating System
Computer Network tutorial	Compiler Design tutorial	Computer Organization and Architecture	Discrete Mathematics Tutorial
Computer Network	Compiler Design	Computer Organization	Discrete Mathematics
Ethical Hacking	Computer Graphics Tutorial	Software Engineering	html tutorial
Ethical Hacking	Computer Graphics	Software Engineering	Web Technology
Cyber Security tutorial	Automata Tutorial	C Language tutorial	C++ tutorial
Cyber Security	Automata	C Programming	C++

Java tutorial

Java

.Net Framework
tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems
tutorial

Control System

Data Mining
Tutorial

Data Mining

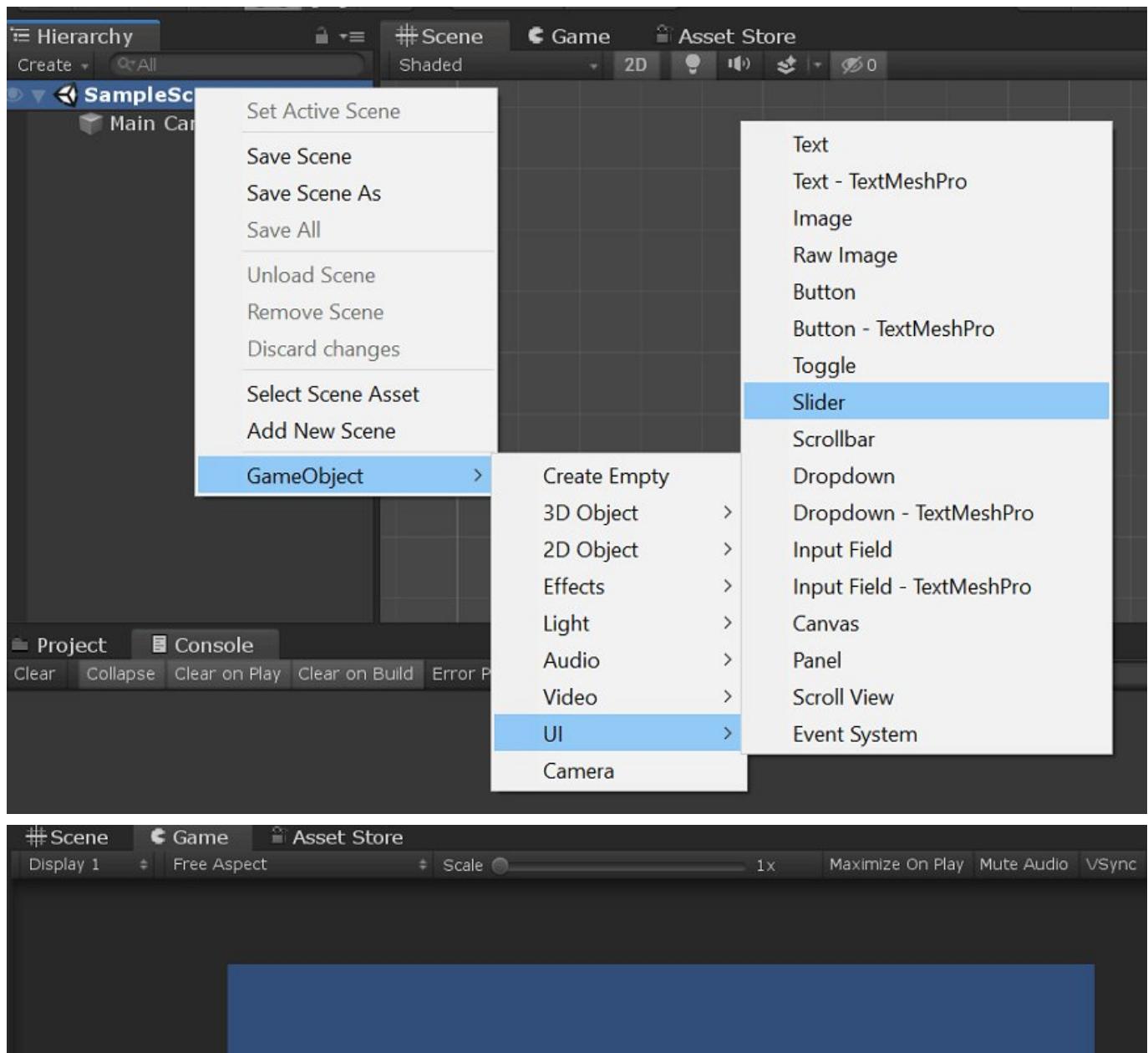
Data Warehouse
Tutorial

Data Warehouse

Unity UI Slider

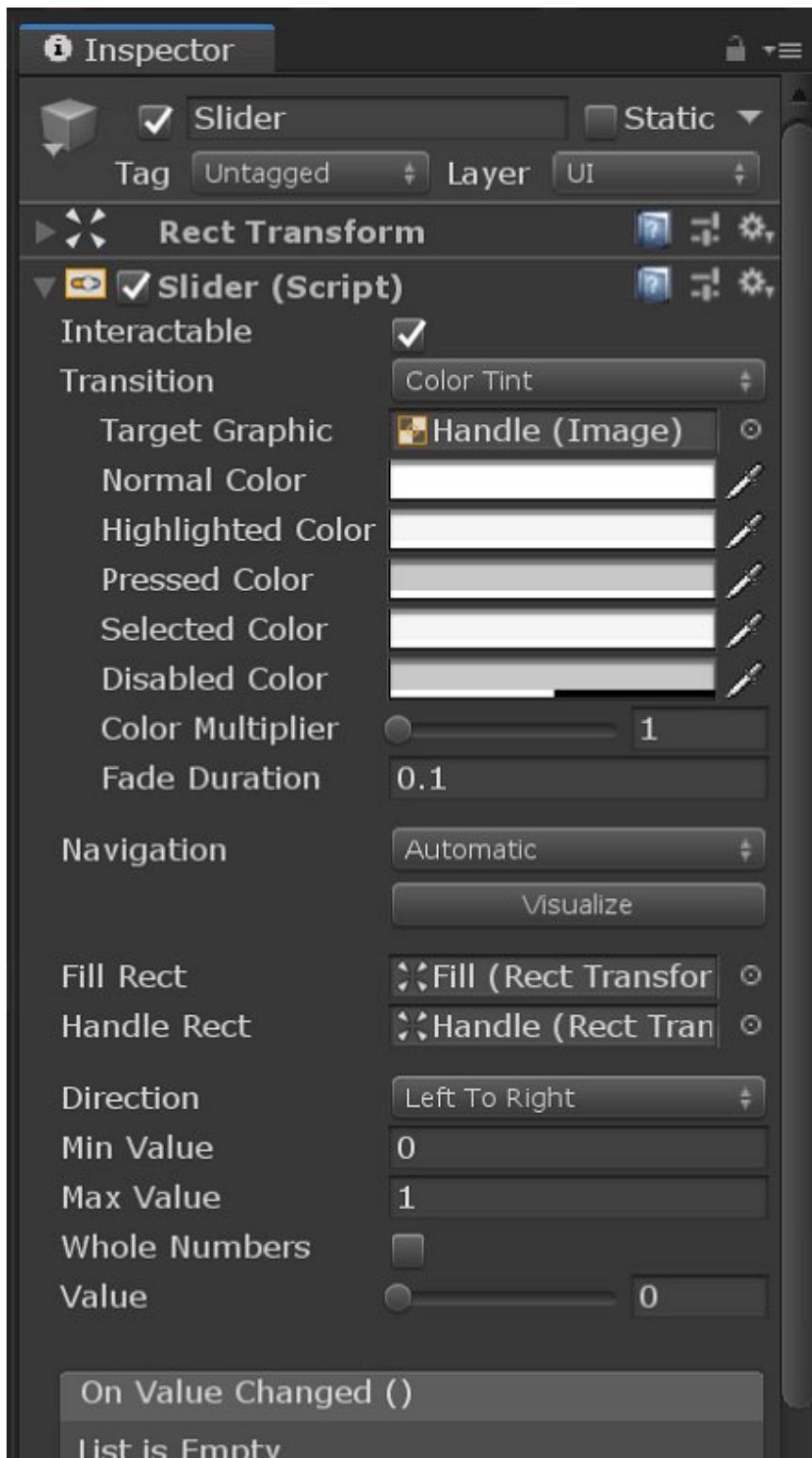
The Slider UI element is commonly used where a **certain value should be set between a minimum and maximum value pair**. One of the most common usages of this is for audio volume, screen brightness, or for doing zoom.

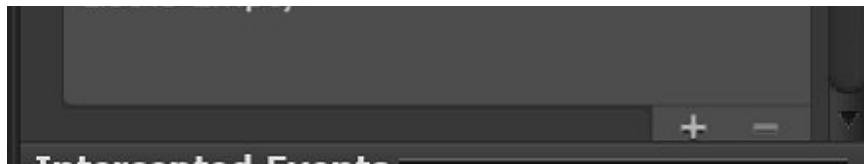
To create a slider UI, right-click on the scene hierarchy and select **GameObject -> UI -> Slider**. A new slider element will display on your scene.





Now, select the slider and go to the Inspector tab.





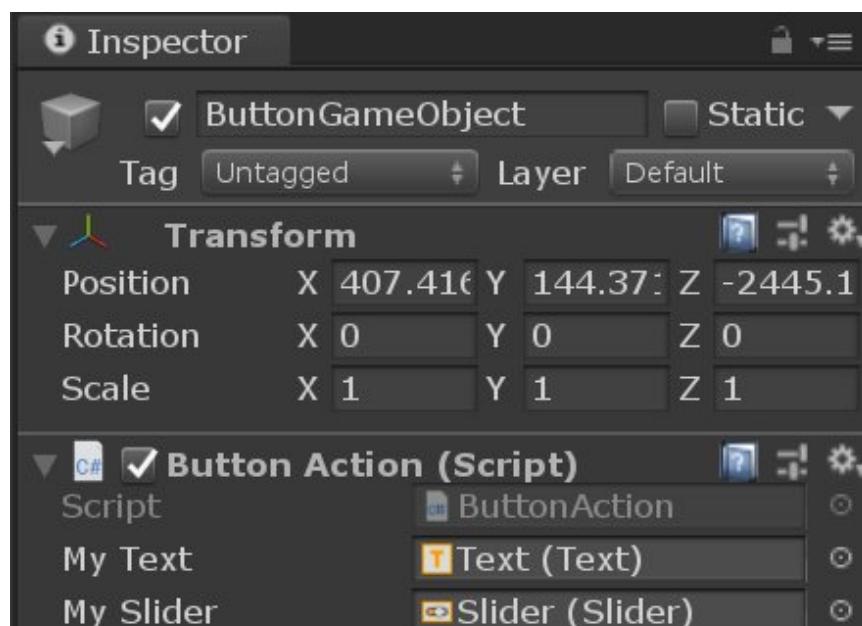
Let's try to make a volume slider. For this open the `ButtonAction` script which we previously created and did some changes on it:

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

public class ButtonAction : MonoBehaviour
{
    public Text myText;
    public Slider mySlider;
    void Update() {
        myText.text = "Current Volume: " + mySlider.value;
    }
}
```

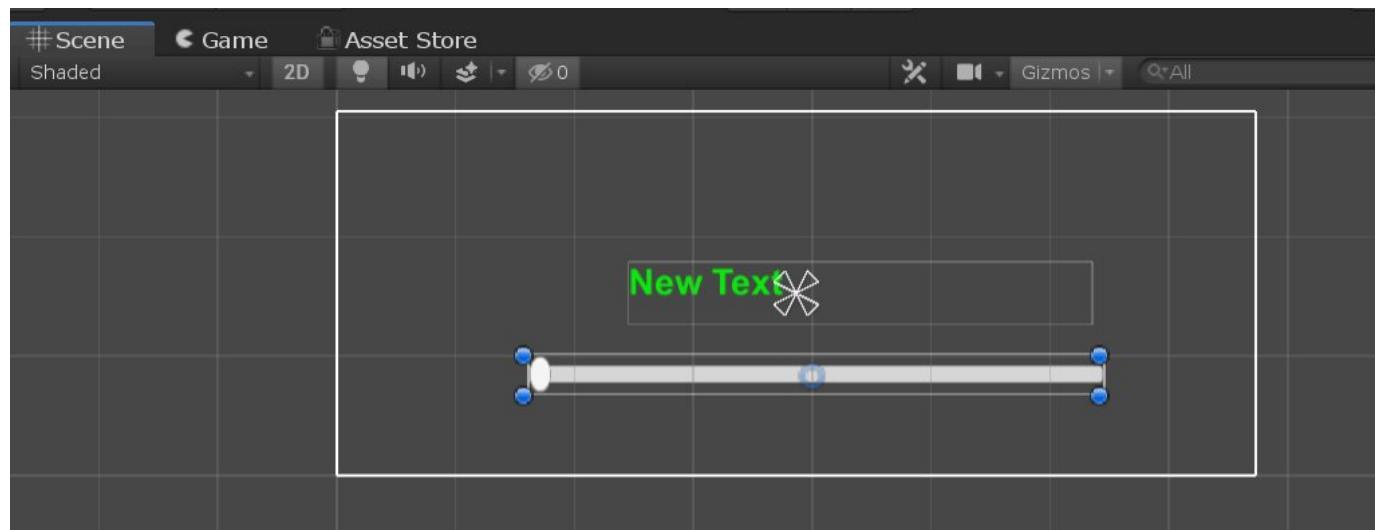
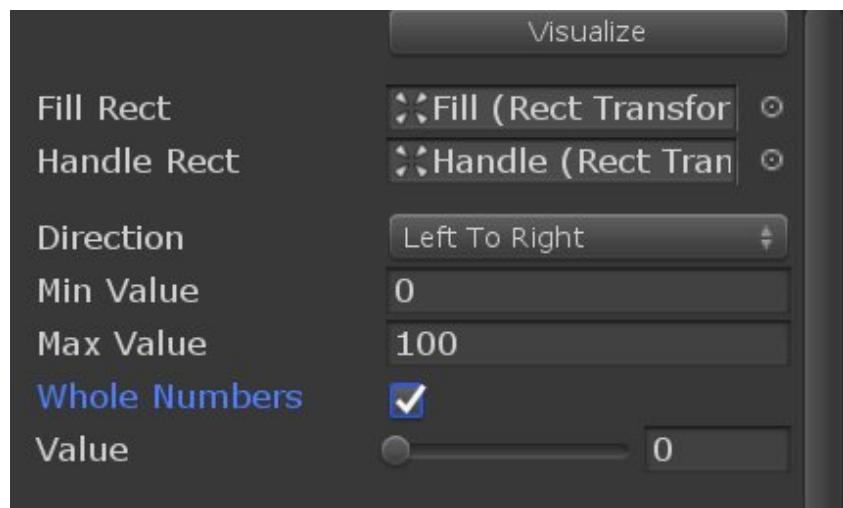
Attach this script to the `ButtonGameObject` (which we previously created). Now, you will get a new slot for slider in the `Inspector` tab of `ButtonGameObject`.

Drag the slider from the `Hierarchy` tab to the newly created slot (`My Slider`) of `ButtonGameObject`.



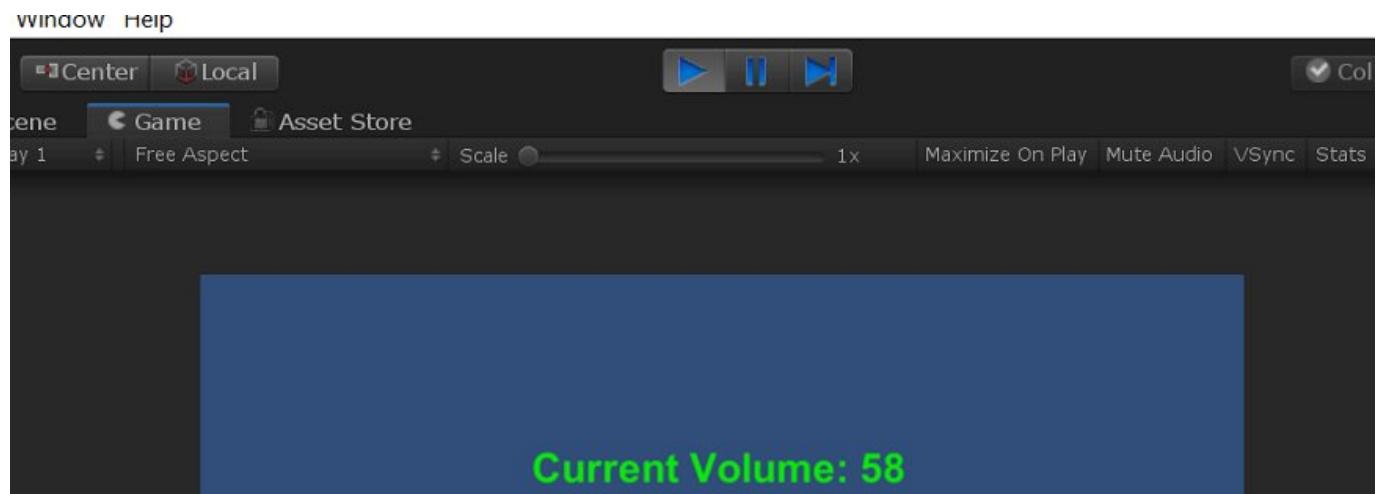
Add Component

Change the slider's maximum value to 100. And check mark for Whole Numbers.



Here, I changed the color of Text to green from its properties, so that it will be more visible.

Now press the play button and slide the slider.



[← Prev](#)[Next →](#)

Youtube [For Videos Join Our Youtube Channel: Join Now](#)

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

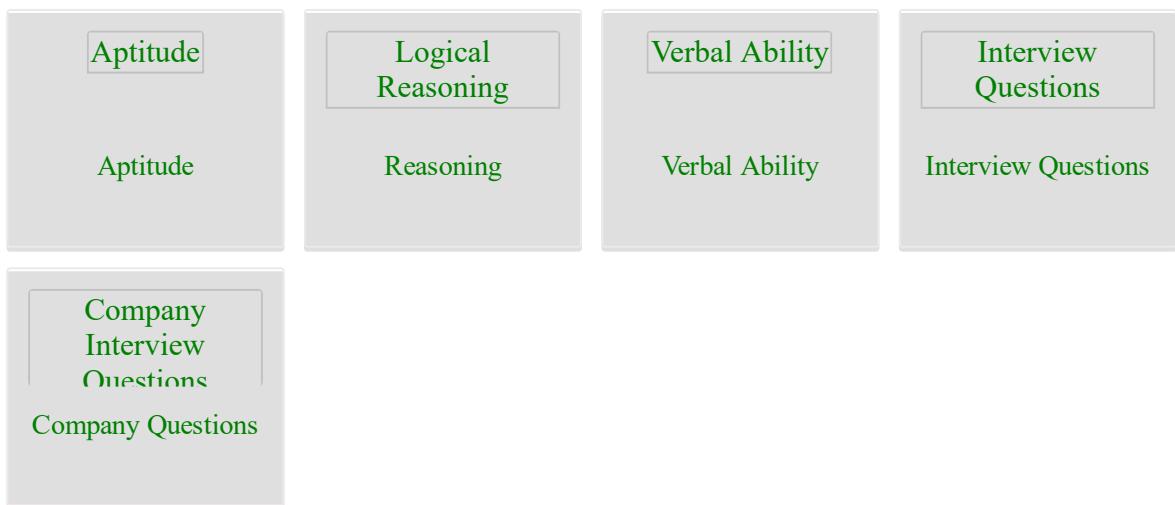


Learn Latest Tutorials

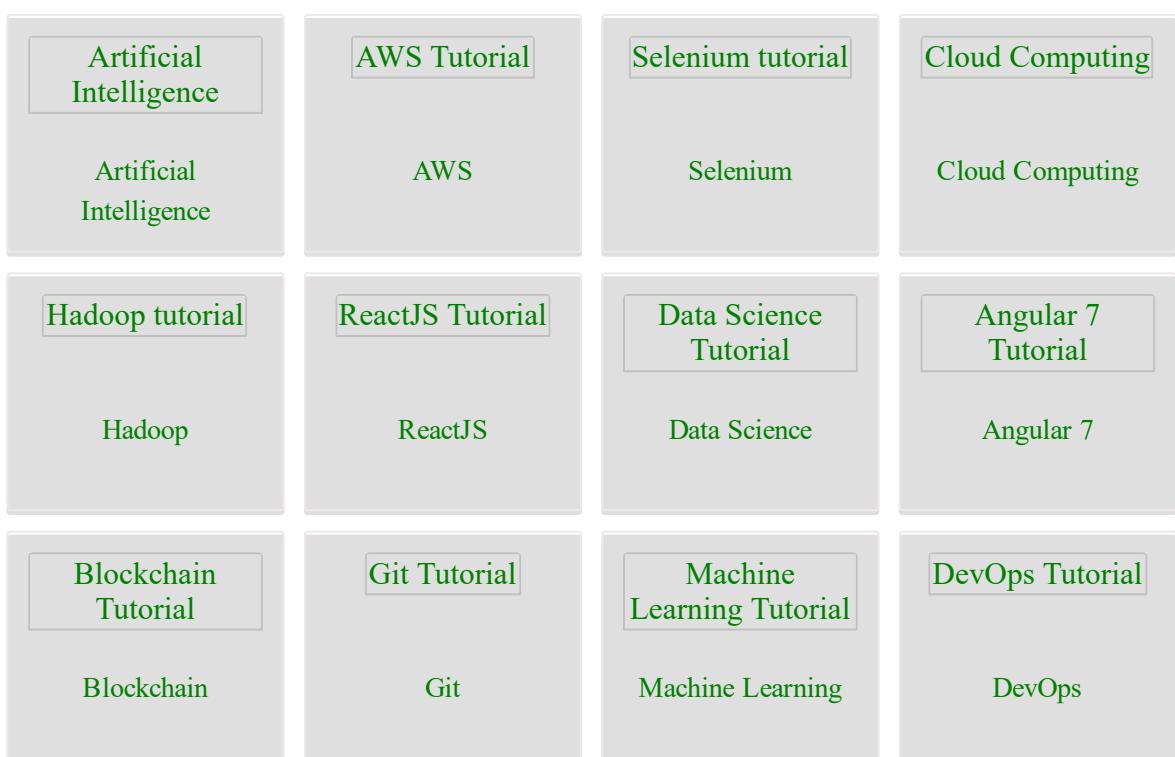
[Splunk tutorial](#)[Splunk](#)[SPSS tutorial](#)[SPSS](#)[Swagger tutorial](#)[Swagger](#)[T-SQL tutorial](#)[Transact-SQL](#)[Tumblr tutorial](#)[Tumblr](#)[React tutorial](#)[ReactJS](#)[Regex tutorial](#)[Regex](#)[Reinforcement learning tutorial](#)[Reinforcement Learning](#)[R Programming tutorial](#)[R Programming](#)[RxJS tutorial](#)[RxJS](#)[React Native tutorial](#)[React Native](#)[Python Design Patterns](#)[Python Design](#)



Preparation



Trending Technologies



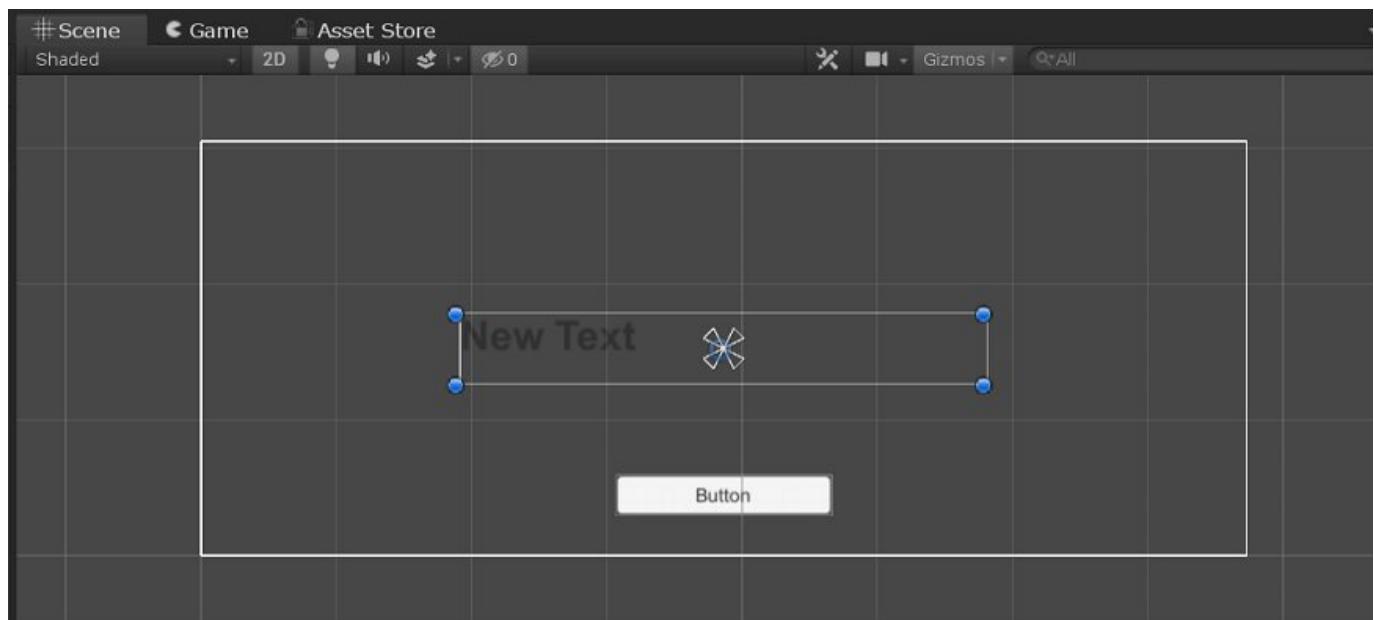
B.Tech / MCA

<p>DBMS tutorial</p> <p>DBMS</p>	<p>Data Structures tutorial</p> <p>Data Structures</p>	<p>DAA tutorial</p> <p>DAA</p>	<p>Operating System</p> <p>Operating System</p>
<p>Computer Network tutorial</p> <p>Computer Network</p>	<p>Compiler Design tutorial</p> <p>Compiler Design</p>	<p>Computer Organization and Architecture</p> <p>Computer Organization</p>	<p>Discrete Mathematics Tutorial</p> <p>Discrete Mathematics</p>
<p>Ethical Hacking</p> <p>Ethical Hacking</p>	<p>Computer Graphics Tutorial</p> <p>Computer Graphics</p>	<p>Software Engineering</p> <p>Software Engineering</p>	<p>html tutorial</p> <p>Web Technology</p>
<p>Cyber Security tutorial</p> <p>Cyber Security</p>	<p>Automata Tutorial</p> <p>Automata</p>	<p>C Language tutorial</p> <p>C Programming</p>	<p>C++ tutorial</p> <p>C++</p>
<p>Java tutorial</p> <p>Java</p>	<p>.Net Framework tutorial</p> <p>.Net</p>	<p>Python tutorial</p> <p>Python</p>	<p>List of Programs</p> <p>Programs</p>
<p>Control Systems tutorial</p> <p>Control System</p>	<p>Data Mining Tutorial</p> <p>Data Mining</p>	<p>Data Warehouse Tutorial</p> <p>Data Warehouse</p>	

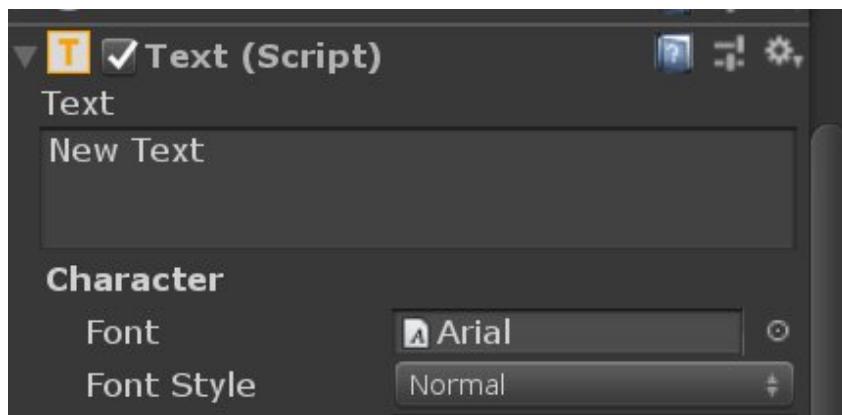
Unity UI Text

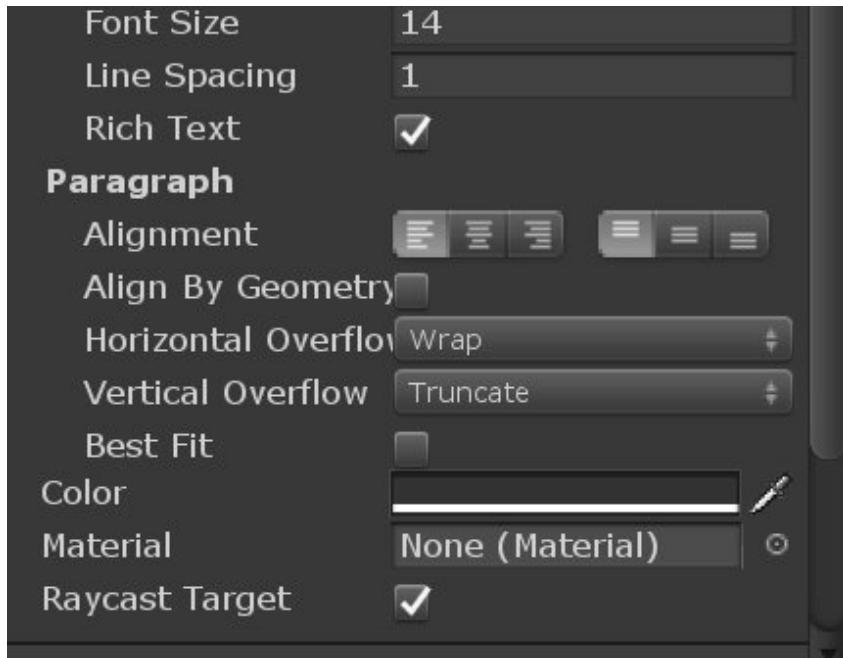
The UI text element displays a non-interactive piece of text to the user. Text elements can be used to provide captions or labels for other GUI controls or to display instructions or other text. For example printing the current score of the player to the screen requires the numeric value of the score to be converted to a string, generally through the `.ToString()` method, before it is displayed.

To insert a Text UI element in Unity, right-click on the Scene Hierarchy, then select GameObject -> UI -> Text.



There are many properties of the Text element. In which Text Field is the most important property. You can type out what you want the text box to show in that field.





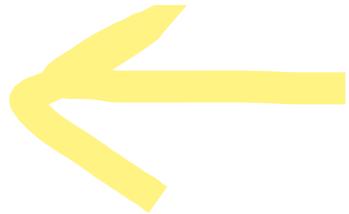
You can change the font of the text; you must first import the font file from your system into Unity as an Asset.

We can also accept the Text element through the scripting; this is where the importance of dynamic UI comes in.

Let's see one simple example, instead of the console, which is printing how many times the button has been pressed, as in the previous chapter; let's print it out on the game screen through the Text element. To do so, open your previous script file (ButtonAction.cs) and make some changes to it:

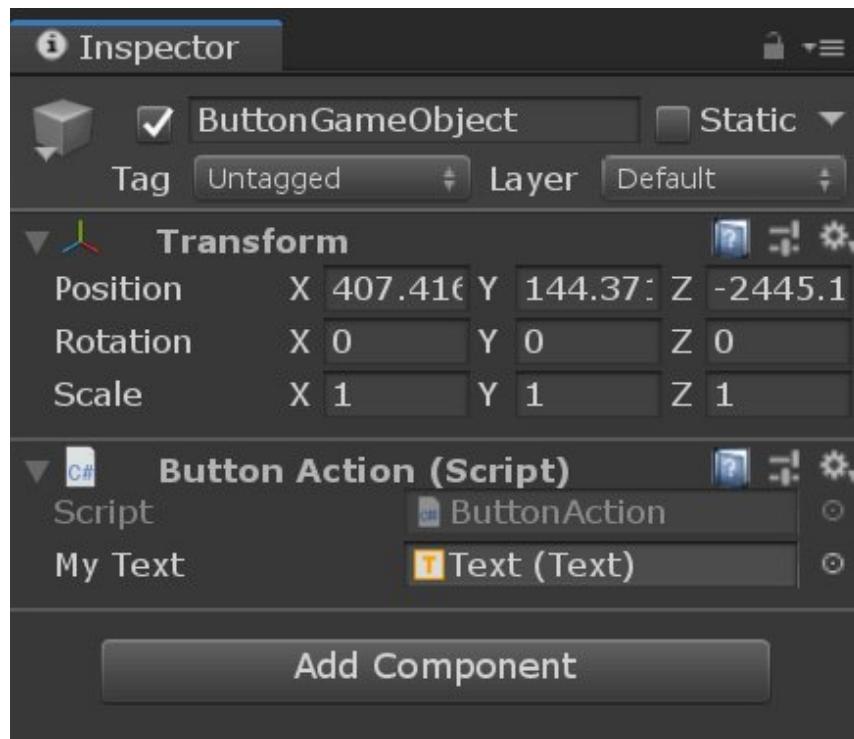
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ButtonAction : MonoBehaviour
{
    int n;
    public Text myText;
    public void OnButtonPress(){
        n++;
        myText.text = "Button clicked " + n + " times.";
    }
}
```

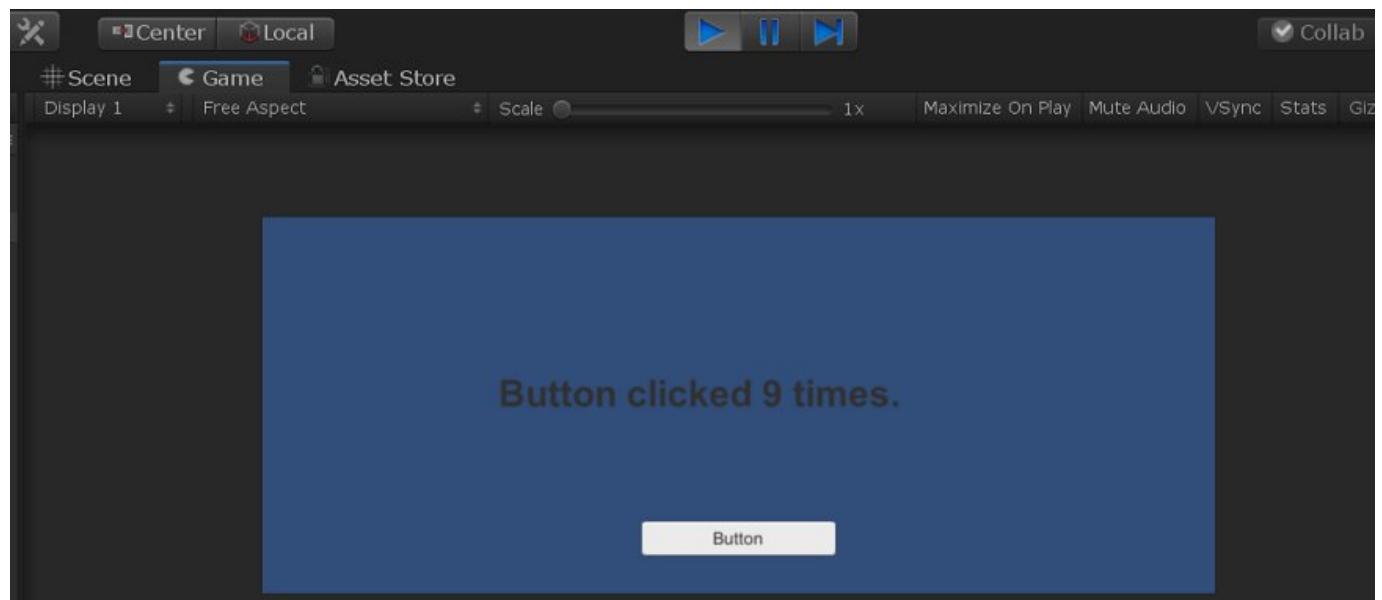


Here, we added a public Text variable, where we can drag and drop our Text UI element.

Save your script and go to the ButtonGameObject, and you will now see the new slot for the Text UI element. Drag and drop your Text GameObject on to the slot.



Now hit the play button.



← Prev

Next →

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share



Learn Latest Tutorials

[Splunk tutorial](#)

Splunk

[SPSS tutorial](#)

SPSS

[Swagger tutorial](#)

Swagger

[T-SQL tutorial](#)

Transact-SQL

[Tumblr tutorial](#)

Tumblr

[React tutorial](#)

ReactJS

[Regex tutorial](#)

Regex

[Reinforcement learning tutorial](#)

Reinforcement Learning

[R Programming tutorial](#)

R Programming

[RxJS tutorial](#)

RxJS

[React Native tutorial](#)

React Native

[Python Design Patterns](#)

Python Design Patterns

[Python Pillow tutorial](#)

Python Pillow

[Python Turtle tutorial](#)

Python Turtle

[Keras tutorial](#)

Keras

Preparation



Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System

Operating System

Computer Network tutorial

Computer Network tutorial

Compiler Design tutorial

Compiler Design tutorial

Computer Organization and Architecture tutorial

Computer Organization and Architecture tutorial

Discrete Mathematics Tutorial

Discrete Mathematics Tutorial

Computer Network

Compiler Design

Architecture

Computer Organization

Tutorial

Discrete Mathematics

Ethical Hacking

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse



ROLES IN GAME DEVELOPMENT

LECTURE 9,10

Modified and presented by
Marwa Al-Hadi

1

OBJECTIVES

- Some points to discuss
- Game development
- Documentation

Part 1: Rules and Mechanics



What are Rules?

- Rules are *formal schemas of game*



CHALLENGE OF DEFINING RULES

1. They **do not need** to be *fixed*(changed in structured ways)
2. They **can** *ignored*(Adding or removing parts from game)
3. They are **not** always *explicit*



HOW TO DESIGN GOOD RULES

● Player must have *meaningful choices*

- Player must be able to **make decisions**
- System must **respond in significant way**

● **Bad Rules:**

● **Guess to pick a winner** because :

- All you can do is **guess** the answer
- Has **no significant effect** on the outcome



INFORMAL VERSUS FORMAL RULES

Informal

- Part of **initial design** process
- Focuses on **how it looks**
- Less concerned with **code**
- Many span **multiple frames**

Formal

- Design is **Part of implementation**
- Links to **code**
- Interactions **link multiple animation frames** together



UNDERSTANDING GAME STATE

● **Spatial means**

- Represent location or state of a game entity
- Also physical values like acceleration

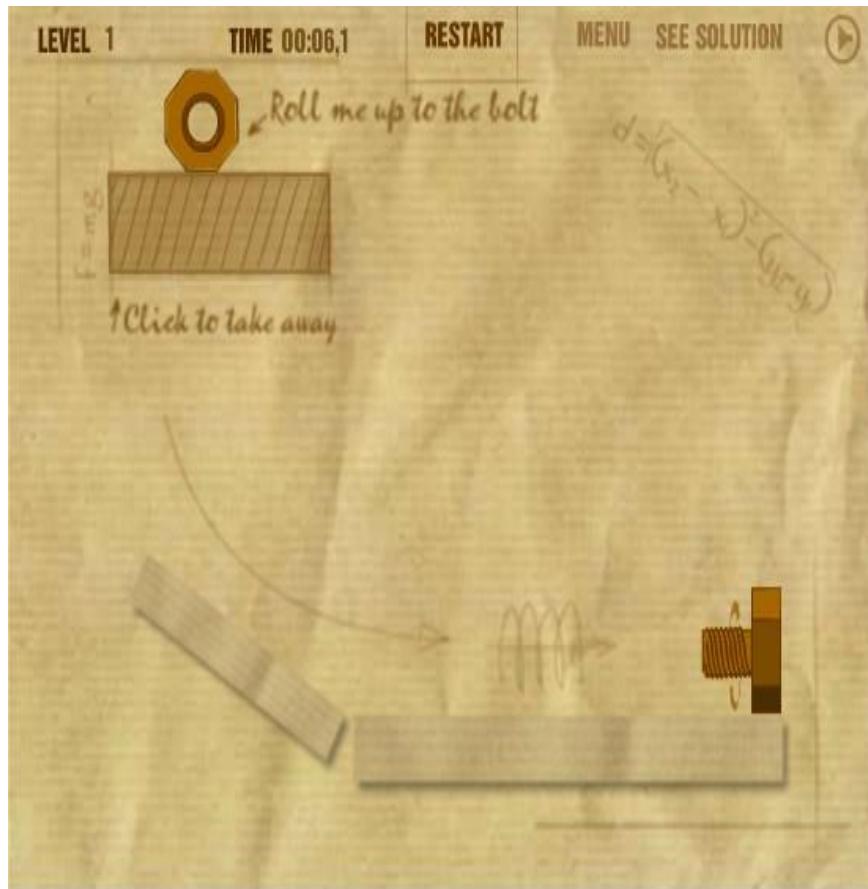


ACTIONS AFFECTING SPATIAL STATE

- Actions Affecting Spatial State in :*movement*
- But there are **many ways to implement**
 - **Direct movement of avatar** (e.g. WASD)
 - A set of **four keys** on a keyboard used as directions to control player characters in certain video games
 - **Indirect** movement of avatar (e.g. pathfinding)
then
 - Alter the **environment**



ALTERING THE ENVIRONMENT



STRATEGY

● **Definition:** sequence of steps in a planed way.

- Allow some *flexibility* in these solution steps
- **Example:** Multiple solutions to Rubik's Cube
- **Example:** Time-rewind in *Braid*
- **Example :***puzzle game.*



11

Rubik's Cube



Time-rewind in *Braid*

RESOURCES AND GAMEPLAY

● Resources:

- Like health, attack, damage
- Expend resources to affect others (e.g. attack)
- May change resources of that entity (e.g. damage)
- are critical in “fighting” mechanics

RESOURCES AND GAMEPLAY

● Resources

- Three basic **categories of resource fighting**
 - **Tug-Of-War:** entities **other** each take
 - **Dot Eating:** **rehtag** ot ecar seititne :*limited* resource
 - **Flower Picking :****rehtag** ot ecar :*unlimited* resource



RESOURCES AND THE GAME ECONOMY

● Sources:

- How a **resource** can increase
- **Examples:** ammunition clips, health packs



● Drains:

- How a **resource** can decrease
- **Examples:** firing weapon, player damage



RESOURCES AND THE GAME ECONOMY

● Converters:

- Changes one resource to another
- **Example:** vendors, *Starcraft* barracks

● Traders:

- Exchange resources between entities
- Mainly (but not always) in multiplayer games



ECONOMIC CHALLENGES

● You can **use resources to**

- Control player progression
- Modify player abilities
- Create a large possibility space
- Create strategic gameplay

ECONOMIC CHALLENGES

- Do not need a lot of resources
- Not every game is a strategy game
- But **almost all** games have some economy



RESOURCES AS DILEMMA

● **Example:** Survival Horror

- Use **ammo** to **shoot** zombie (**Cost**: ammo)
- Use **knife** to **crack** zombie (**Cost**: health)
- Benefit the same in each case



RESOURCES AND MONETIZATION

- some games allow **external sources**
 - Get resources from a friend on Facebook
 - Pay for resources with a credit card



RESOURCES AND MONETIZATION

● *Monetization is*

- Free-to-play, pay-for-stuff
- Modern **business** model for **online games**
- But **BIG objection right now**



EMERGENT BEHAVIOR

● Coupled Interactions

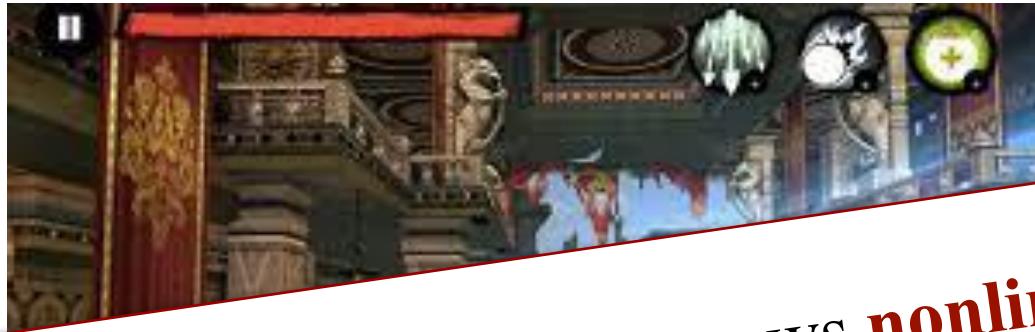
- Two mechanics that can happen at once
- **Verbs:** jump AND run in a platformer
- **Resources:** warrior AND archer in an RTS



EMERGENT BEHAVIOR

● Context-dependent Interactions

- Mechanics combine to give new behavior
- Verbs: jump and run is new form of movement
- Resources: warriors make wall to cover archers



Advantage: game complexity grows **nonlinearly**



EMERGENT ACTIONS



COMMON SPATIAL INTERACTIONS

Collisions-crash-

- Can effect *resources*
 - Player takes damage
 - Player gains power-up
 - Player transfers gold
- Can effect *spatial values*
 - Bounce off collision point
 - Swing from attached rope
 - Attraction to magnet/charge

Detection

- Examples:
 - Line-of-sight
 - Spatial proximity
- Can have *direct* effects
 - Alarms in a stealth game
- Can have *indirect* effects
 - Tower defense targeting

RESOURCE-SPATIAL INTERACTIONS



Spatial Affects Resources

- Resources made by entities
 - Have a spatial location
 - **Ex:** Time to transfer resources
 - **Ex:** Sources be captured
- Resource values are entities
 - Take up physical volume
 - Need space to acquire
 - **Ex:** Inventory in *Deux Ex*

COUPLING IS NOT ENOUGH

- Example of *trivial* coupling:
 - RTS with single unit type – warrior
 - Coupling can arise from multiple warriors
 - When **attack, count number** on each side
- Group of warriors **is** sum of its parts
 - Just make a **single warrior stronger**
 - Discover from *resource analysis*
- Emergent behavior must couple *nonlinearly*
 - If n base mechanics, more than $O(n)$ behaviors



EXAMPLE: *STARCRAFT*

- **Basic units can**
 - Attack in sky and/or land
 - Defend in sky and/or land
 - How can these **combine**?
- **Further complexity:**
 - “Buff” friendly units
 - “Control” enemy units
 - How does this affect game?
- **Challenge** si tahW :
a rof ytixelpmoc laminim
?STR doog



SUMMARY

- Rules are **formal systems** defining your game
 - Specify to change the game state over a single frame
 - Challenge is matching them to your informal design
- **Resources** create *strategic* gameplay
 - Resources define the game economy
 - Strategy is just players making economic choices
- **Interactions** facilitate *emergent behavior*
 - Coupled actions/interactions creating new features
 - Can provide deep, nonlinear complexity



GAME ENGINES

- **Game Engines :**
- are **software development environments** with pre-built gaming components that game developers can use to plan and build interactive video game frameworks for PCs, consoles, mobile devices, and more.



DOES UNITY USE PYTHON???

- No.
- You can write scripts for Unity in C# or JavaScript,
- **the engine and tools aren't written in Python.**
- but There is a plugin to allow you to write Python code (Python for Unity),



WHICH ONE IS BETTER UNREAL ENGINE OR UNITY

- **Graphics:**
- **Both** tools produce **excellent** graphics,
- but **Unreal Engine** is to go-to for the **highest quality** graphics and environmental details.
- **Unreal** supports **faster** enduring than **Unity**, which can speed up post-processing and game development.

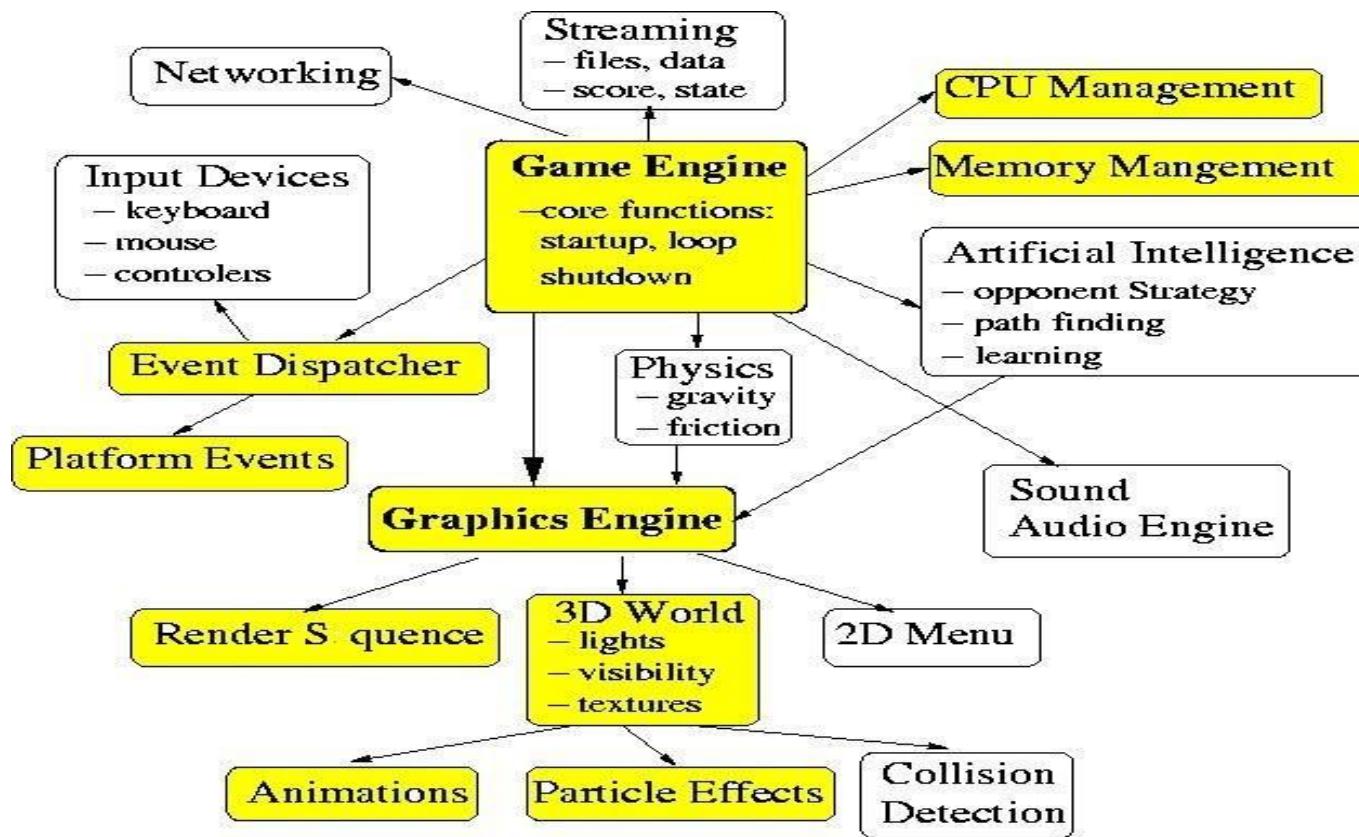


UNITY VS. UNREAL

	Unreal Engine	Unity
Engine Type	Cross-platform	Cross-platform
Developed by	Epic Games	Unity Technologies
Programming Languages	C++ for development	C# ,js for development
Usage	Develop games for PCs, mobiles, consoles, and more	Develop games for PCs, mobiles, consoles, and more
Features	A robust multiplayer framework, VFX, and particle simulation	2D,3D improvements, animation, creating snapshots
Source Code	Open-source	Not open-source.
Pricing	Free	Basic version is free
Learning Curve	Difficult to learn	Easy to learn with an intuitive interface
Graphics	Photorealistic graphics	Good overall graphics, but less refined than Unreal



GAME PROGRAMMING



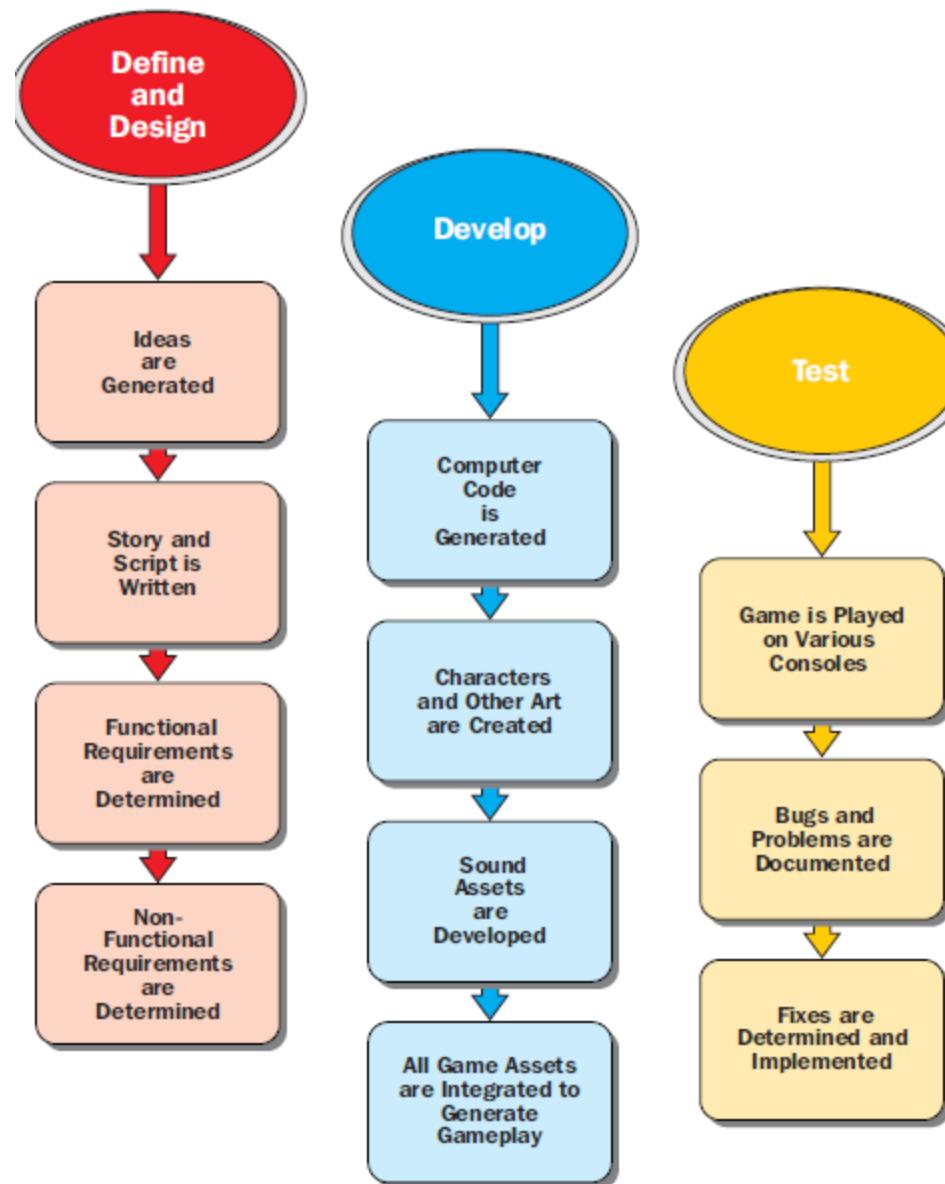
ANTICIPATED PROBLEMS

- What are the three stages of game development?

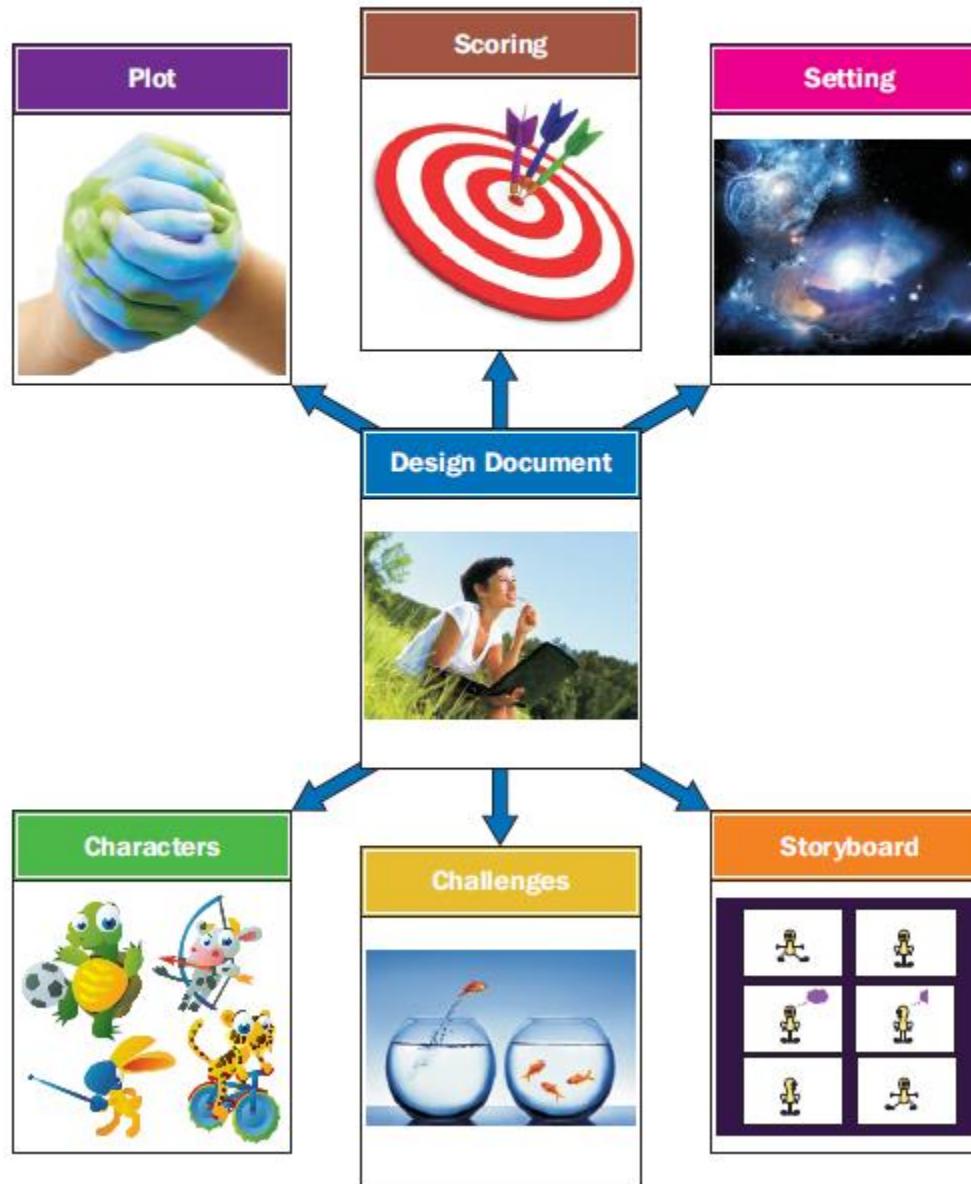
Three stages of game development

- **A. Stage 1: Define and design game.**
- **B. Stage 2: Define game assets and develop game**
- **C. Stage 3: Game testing**

THREE STAGES OF GAME DEVELOPMENT

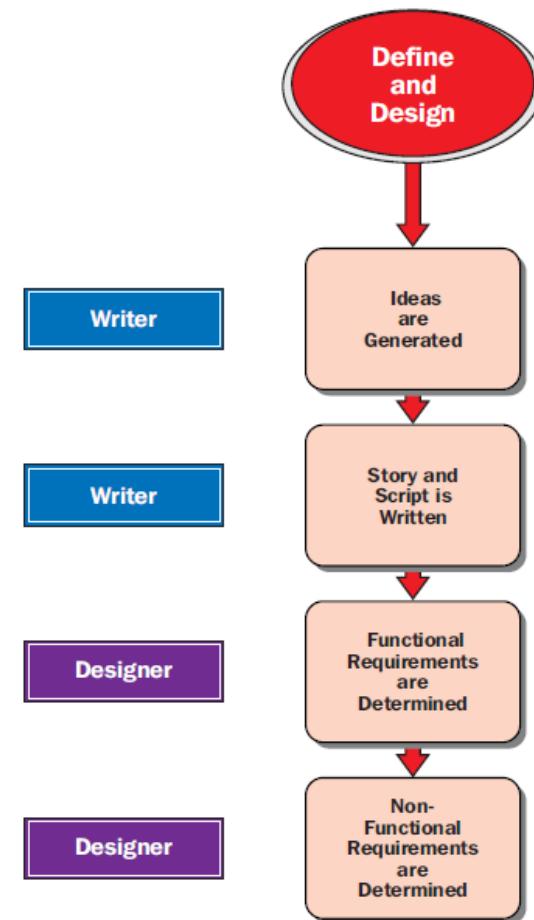


COMPONENTS OF A DESIGN DOCUMENT



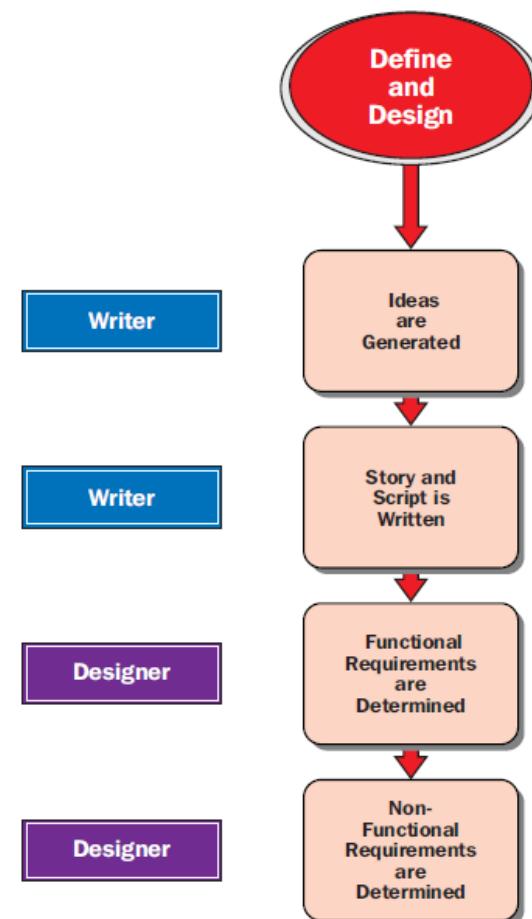
responsibilities associated with each stage of game development?

- A. **Define and design**(writer and design)
 - 1. A *writer* is a person who *writes* the *storyline* and *script* for an electronic game.
 - a. A *design document* is created during this stage.
 - b. *Game play* describes every experience that a player may encounter while playing the game.



responsibilities associated with each stage of game development?

- A. **Define and design**
- 2. A *game designer* is a person who helps determine if the game, as written in the design document, can be or should be produced.

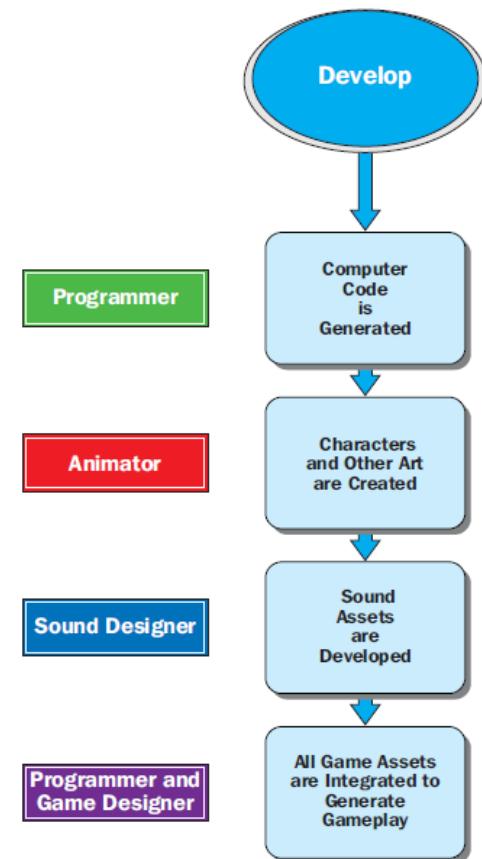


responsibilities associated with each stage of game development?

- **B. Develop**

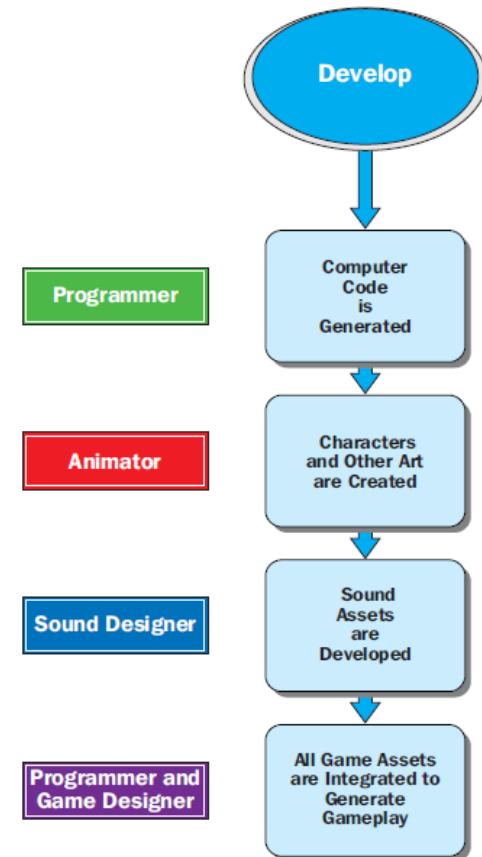
- 1. An *animator* (artist)

- creates **characters** and
- Create **character movements** (**animations**) as well as **environment assets** (all of the artwork that makes up the setting in which the game is played) that will appear in the game.



responsibilities associated with each stage of game development?

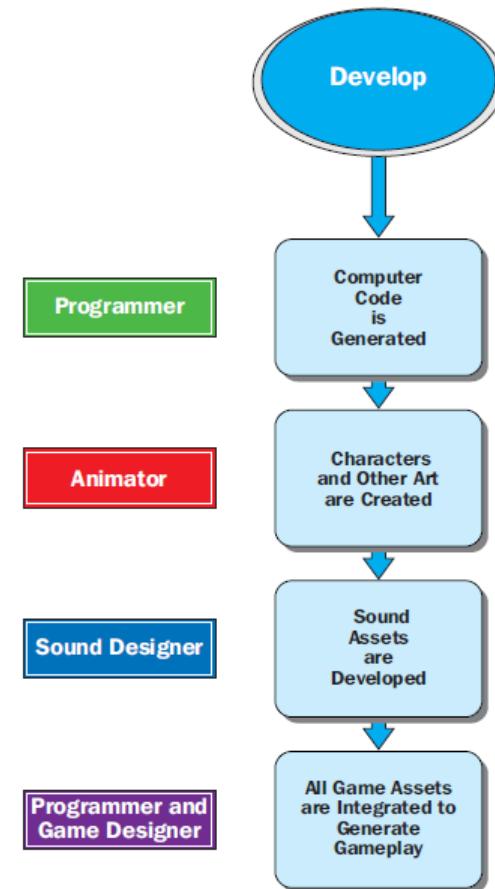
- B. Develop
- 2. A *programmer*
 - writes the code that causes the interaction between game assets and determines game play.



responsibilities associated with each stage of game development?

○ 3. A *sound designer*

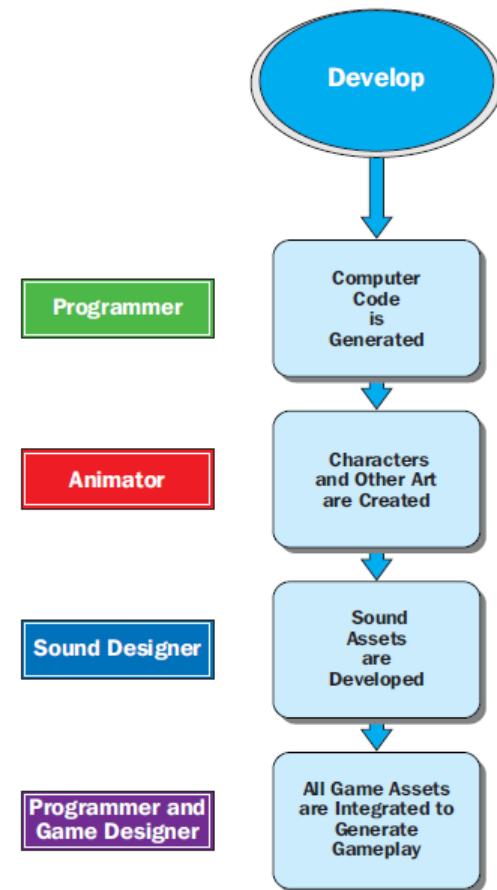
- creates sound effects and music, which add to the game play experience.



responsibilities associated with each stage of game development?

○ 4. A *producer* (director)

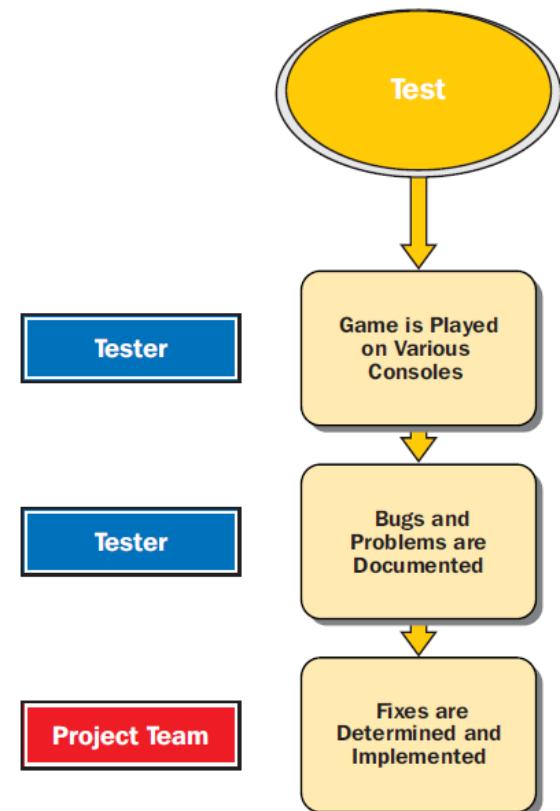
- acts as a **project manager** to be
 - sure that the development team is working together to
 - **meet design requirements** and deadlines within budget constraints.
- **leads** the ***project team***, which consists of all individuals involved in the design, development, testing, and final outcome of the game.



positions and responsibilities associated with each stage of game development?

o C. Test—

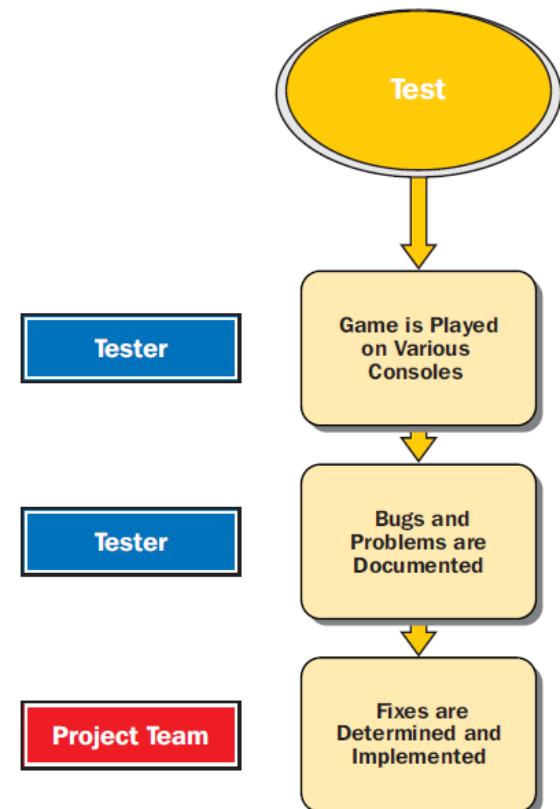
- During the test phase, a *tester* plays each component of the game repeatedly in an attempt to
 - find problems or
 - to break the game.



positions and responsibilities associated with each stage of game development?

o C. Test—

- Testers must effectively communicate with the project team so the problems can be fixed



What is Game Testing??????

- ▶ Subset Of Game Development.
- ▶ Software Testing process for Quality Control Of Video Games.



Your speed, passion and dedication towards the QA work even on such low budget project is remarkable. I really appreciate your services.

Indie game developer, India

More About Game Testing.!!

- ❖ Its said that “All work and no play makes Jack a dull Boy”.
But what if you get paid to play at work?
- ❖ Yes it is possible you will be paid at least 7-20 US \$ for an hour..
- ❖ Wicked isn't it??

Play ➤ Trace

Document

Report ➤ Yes Get
Paid!

So What Do Game Testers Actually Trace??



So What Do Game Testers Actually Trace?? Contd....

- ▶ The primary function of game testing is the discovery and documentation of Software Defects (aka bugs).
- ▶ So That's it they trace “bugs”.

What is a bug?? It Sounds So yucky!!

- ▶ Bug is not a worm for sure.. Yeah a Bug are uncovered errors that are bound to creep in during the development of any software Or a Game..
- ▶ These may range from bugs to art glitches to logic errors and level bugs.

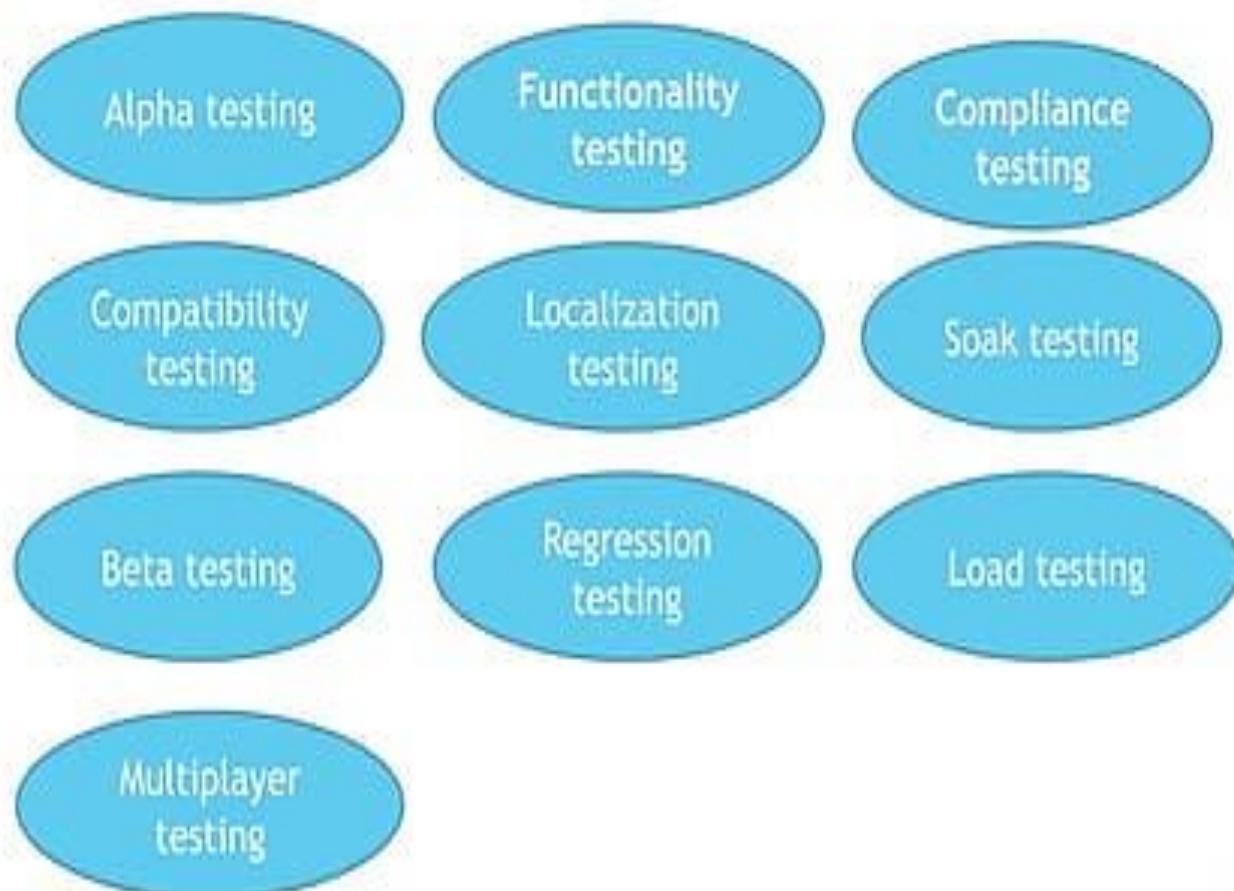


What is a bug?? It Sounds So yucky!!.

Contd..

- ▶ Bugs do have Categories
- ▶ A bugs are critical bugs that prevent the game from being shipped, for example, they may crash the game.
- ▶ B bugs are essential problems that require attention, however the game may still be playable. Multiple B bugs are equally severe to an A bug.
- ▶ C bugs are small and obscure problems, often in form of recommendation rather than bugs.

Categories Of Testing



Look Out Some Bugs Already crept in our
mind...

Misconceptions About Game Testers & Game Testing

- ▶ Its damn easy money out there just stick your nose to the Game and earn a lot.
- ▶ Game Testing has a low job profile.
- ▶ Game Testing is different from Software testing.
- ▶ They have less importance then the developer who has actually coded the Game.
- ▶ Its easier then any Programming languages as c/c++/c# or java.



Skills Required??

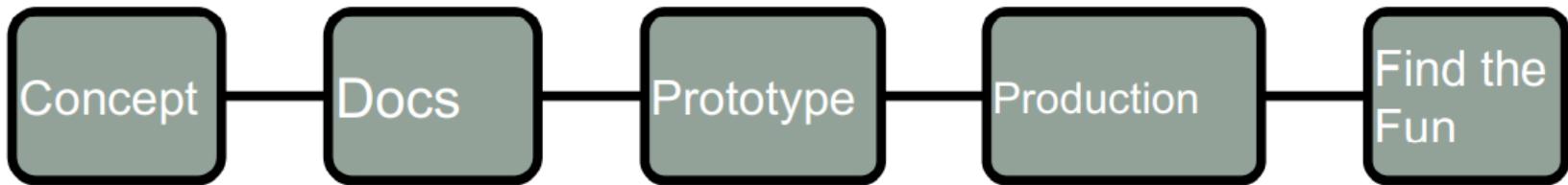


Technical Skills that Will Get You Ahead in the Gaming Industry !!!

- ▶ Fundamental Love and Passion for Games.
- ▶ Eye for Details.
- ▶ Critical Thinking.
- ▶ Ability to stay Focused throughout.
- ▶ Team Work Skills.
- ▶ Programming Skills
- ▶ Data Entry and Processing.
- ▶ Technical Writing.
- ▶ Math and Logic.
- ▶ Degree- Not Required.

DOCUMENTATION

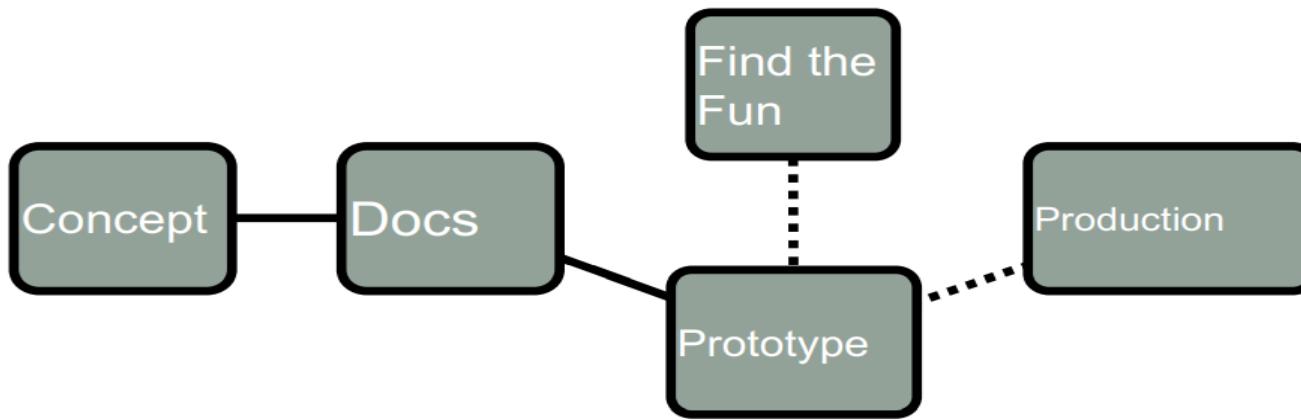
TRADITIONAL PROCESS



Traditional PROCESS



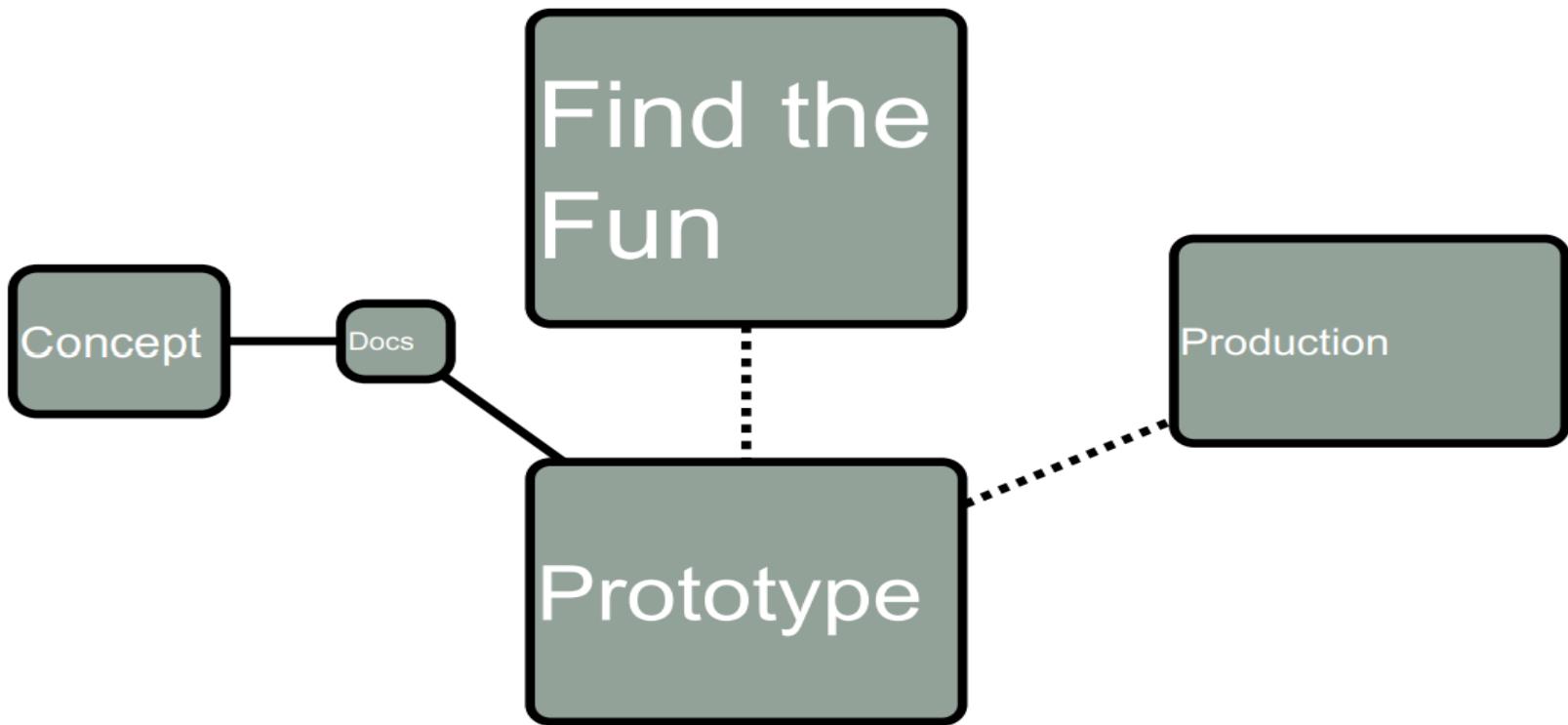
REVISED PROCESS



Revised PROCESS



LAST VERSION OF PROCESS



DESIGN DOCUMENTS

1. Game Mechanics

1. Core Gameplay
2. Game Flow
3. Characters/Units
4. Gameplay Elements
5. Game Physics
6. Statistics
7. AI
8. Multiplayer

2. User Interface

1. Flow chart
2. Functional Requirements
3. Mock-up
4. Buttons, icons, pointers

3. Art and Video

1. Goals, style, mood
2. 2D art and animation
 1. GUI
 2. Special Effects
3. 3D art and animation
4. Cinematics

4. Sound and Music

1. Goals, style, format
2. Sound effects
 - 1.GUI stceffe laicepS.4.2.2
 - 2.tnemnorivnE.4.2.3
3. Music
 - 1.Events
 - 2.System screens
 - 3.Level theme
 - 4.Situations
 - 5.Cinematic soundtrack

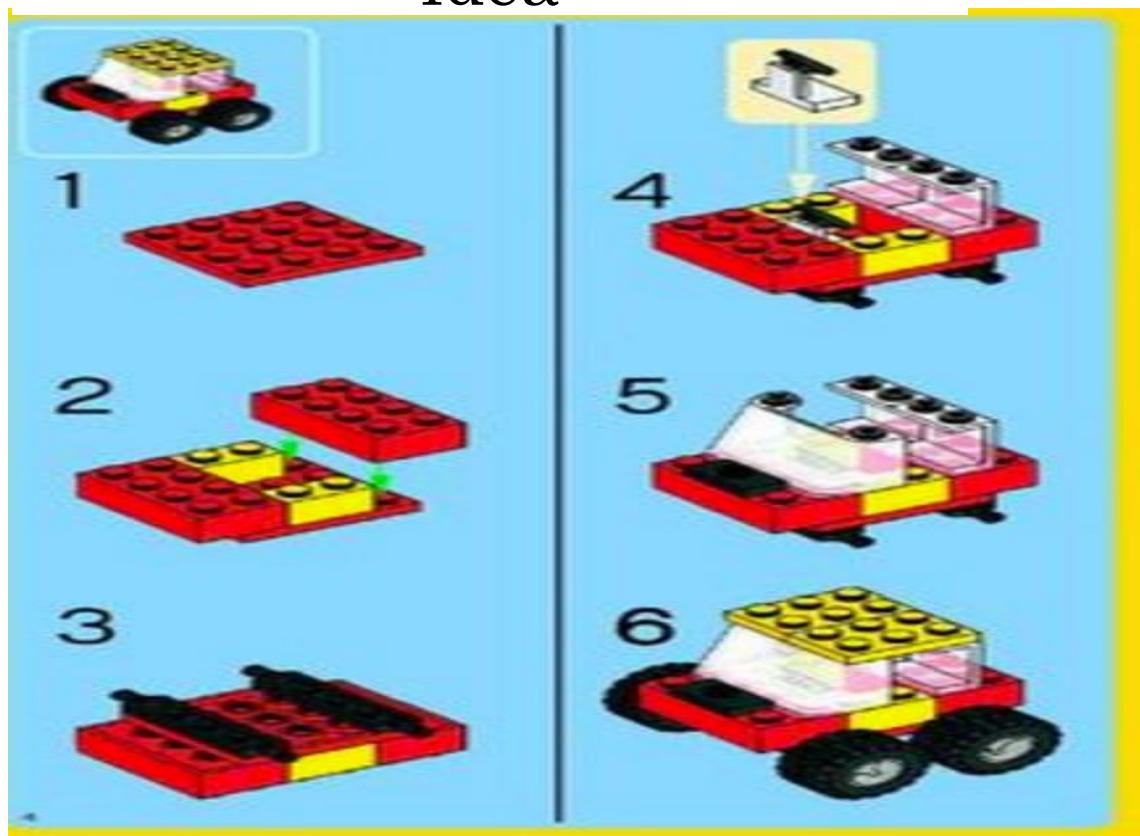
5. Story

1. Backstory and world
2. Character descriptions
3. Game text, dialog requirements
4. Sample scripts

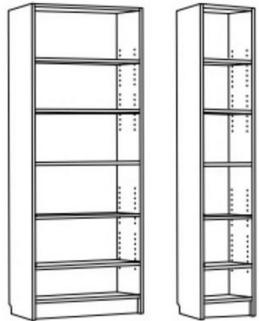
6. Level Requirements

1. Level Diagrams
 - 1.Flow diagrams
2. Asset revelation schedule

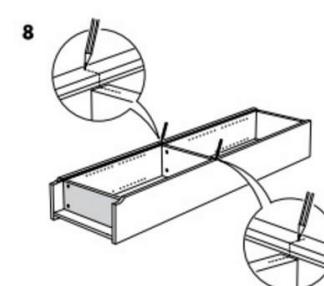
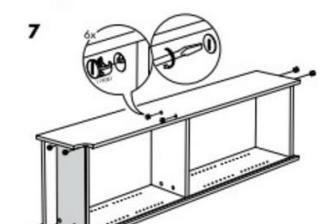
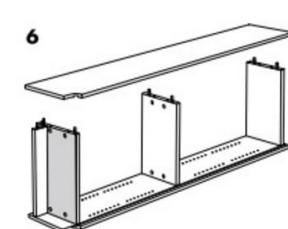
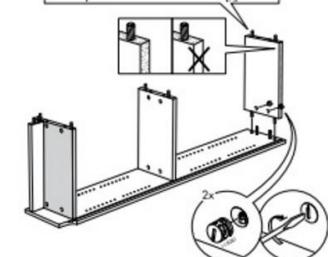
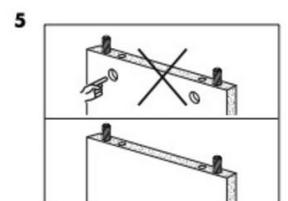
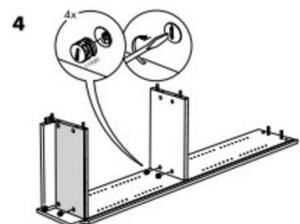
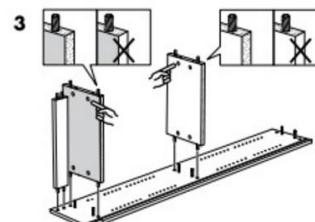
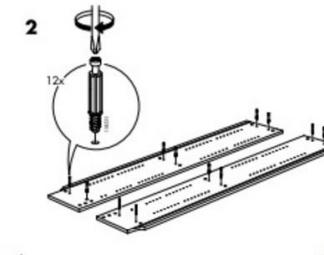
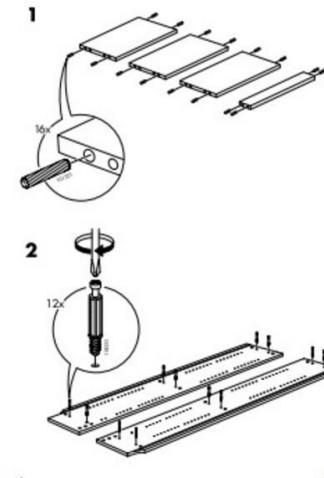
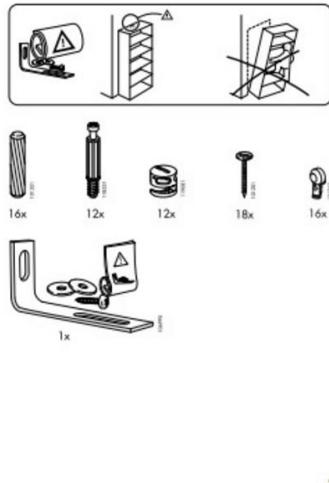
idea



BILLY

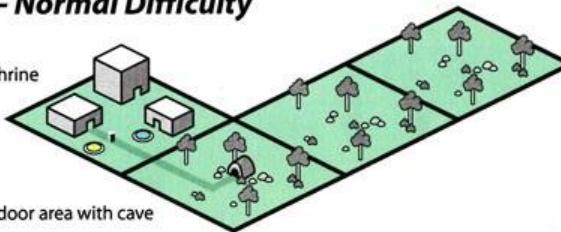


IKKE
Design and Quality
from Sweden



The World of DHack - Normal Difficulty

Small town with healing shrine and identify shrine.

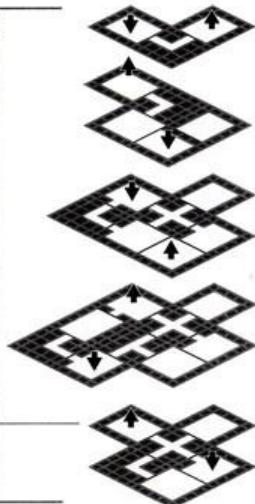


Outdoor area DRLG
(This is for demo purposes and can be any size.)

Small outdoor area with cave

Mines (Levels 1 - 5)

It was once a prosperous mining operation but now it lies in ruins. What caused this destruction?



Level 1 - Mines

(Up and down stairway plus 1 random tile)

Level 2 - Mines

(Up and down stairway plus 3 random tiles)

Level 3 - Mines

(Up and down stairway plus 5 random tiles)

Level 4 - Mines

(Up and down stairway plus 7 random tiles)

Level 5 - Mines

Swarm Level
(Up and down stairway plus 4 random tiles)

Swarm Level

Every 10th level, starting with the 5th, contains only one type of weak monster in large numbers.

Caves (Levels 6 - 10)

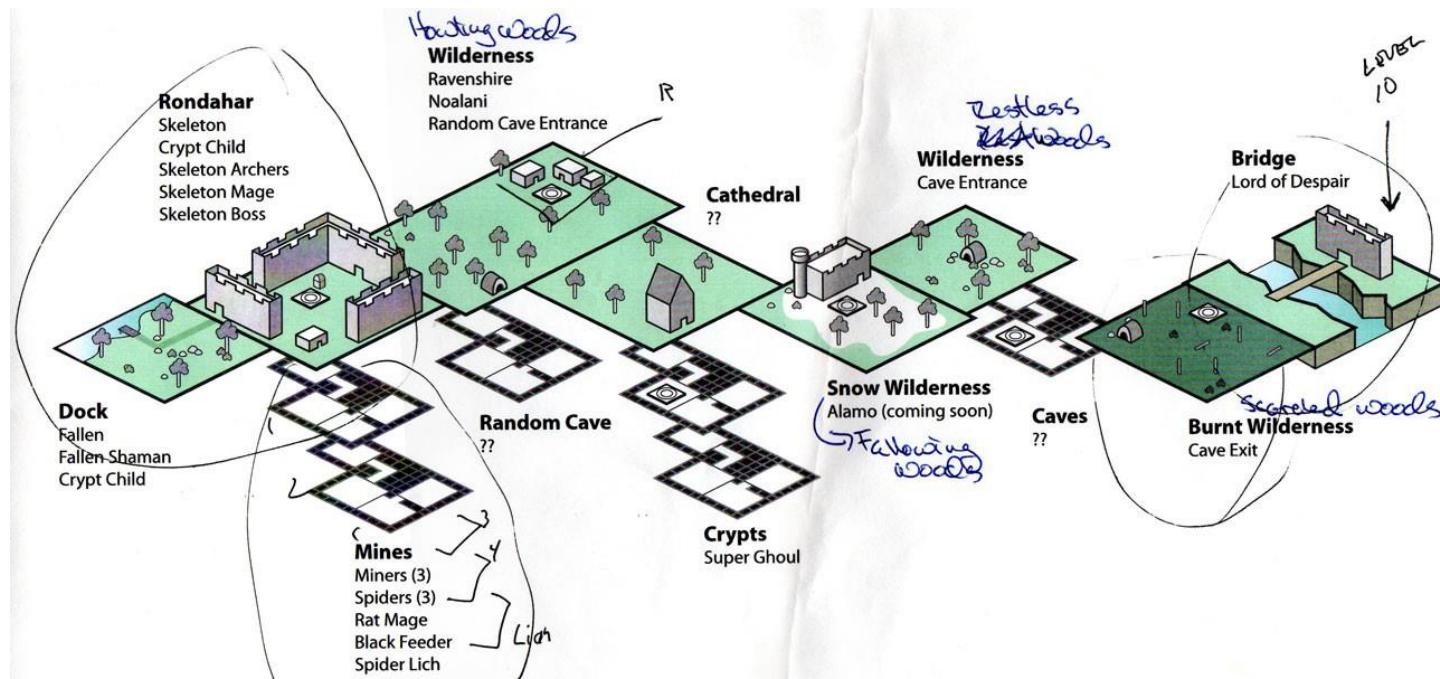
The first group of miners that broke through to this vast underground cave system unleashed horrors beyond comprehension.

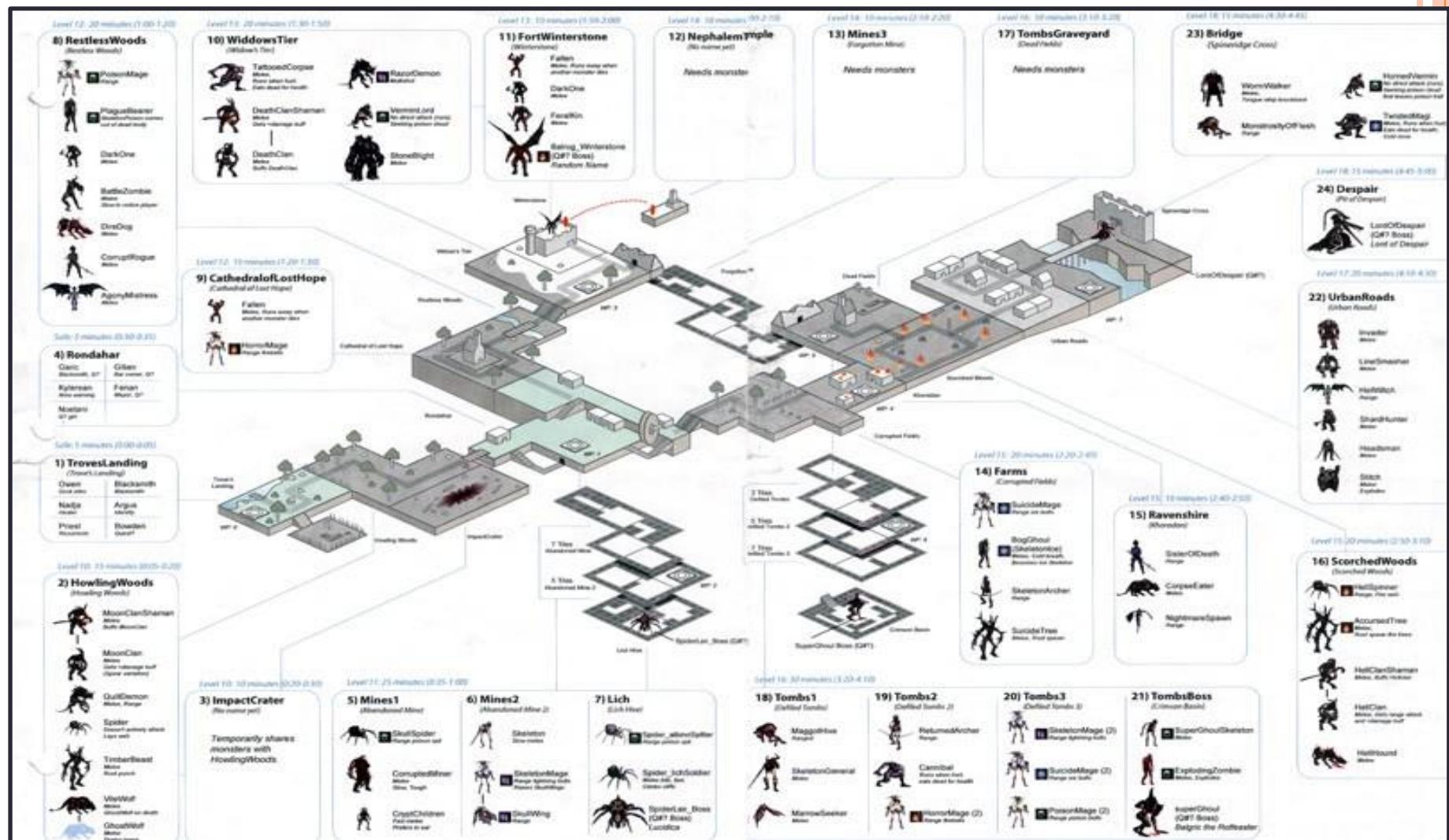


Level 6 - Caves

(Up and down stairway plus 1 random tile)







Level 13: 20 minutes (1:30-1:50)

10) WiddowsTier



TattooedCorpse
Melee,
Runs when hurt,
Eats dead for health



DeathClan
Melee
Buff's DeathClan



DeathClanShaman
Melee
Gets +damage buff



RazorDemon
Range lightning



VerminLord
No direct attack (runs),
Seeking poison cloud



StoneBlight
Melee

Level 13: 10 minutes (1:50-2:00)

11) Winterstone



Fallen
Melee, Runs away when
another monster dies



DarkOne
Melee



FeralKin
Melee



Balrog_Winterstone
(Q#? Boss)
Random Name

Level 14: 20 n

12) Neph

Needs n

1

rel 12: 10 minutes (1:20-1:30)

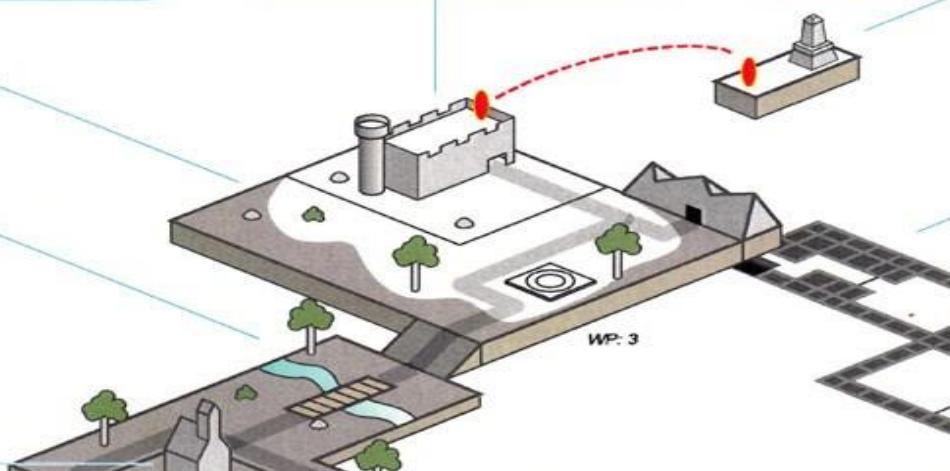
9) CathedralofLostHope

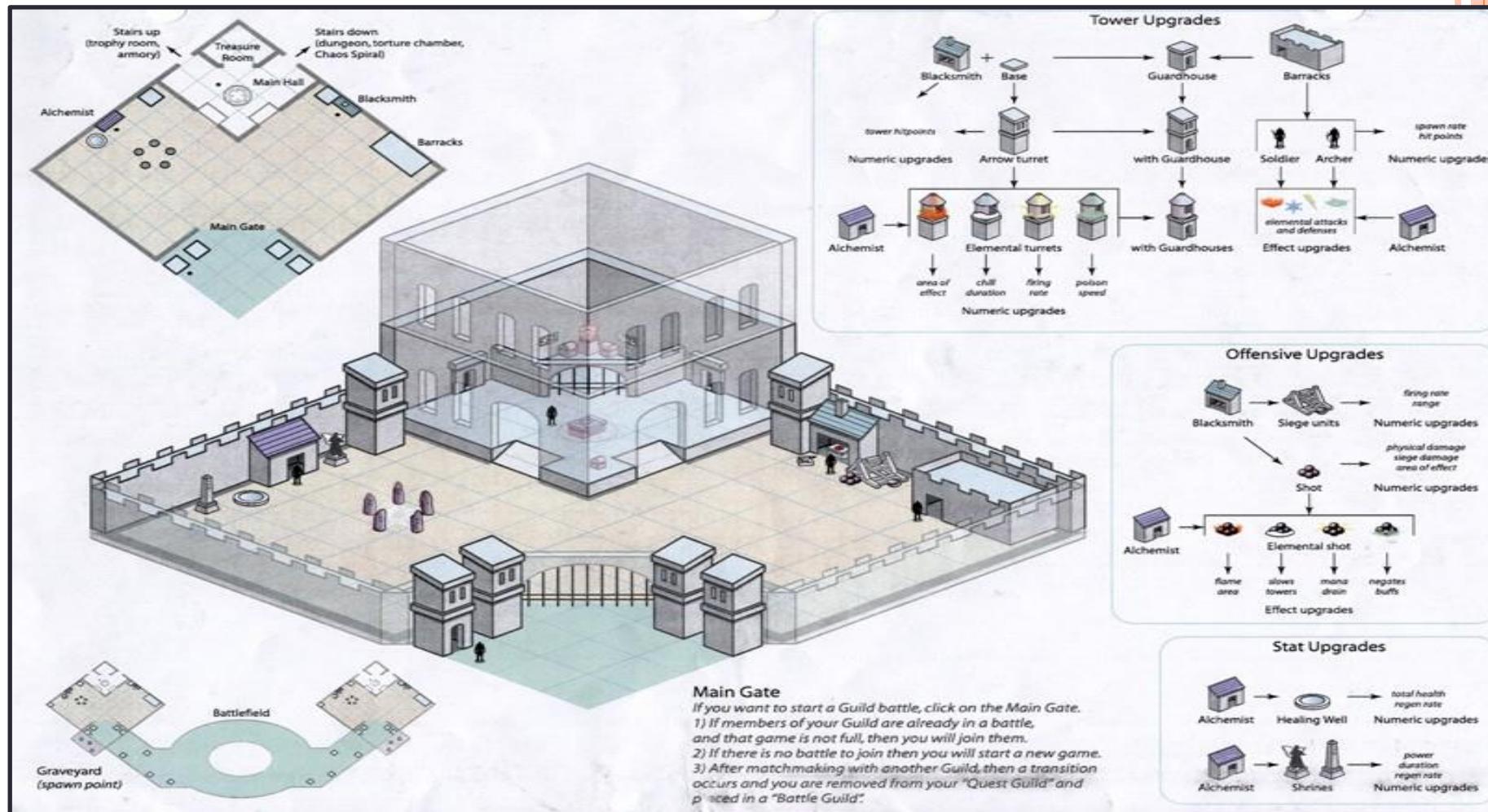


Fallen
Melee, Runs away when
another monster dies

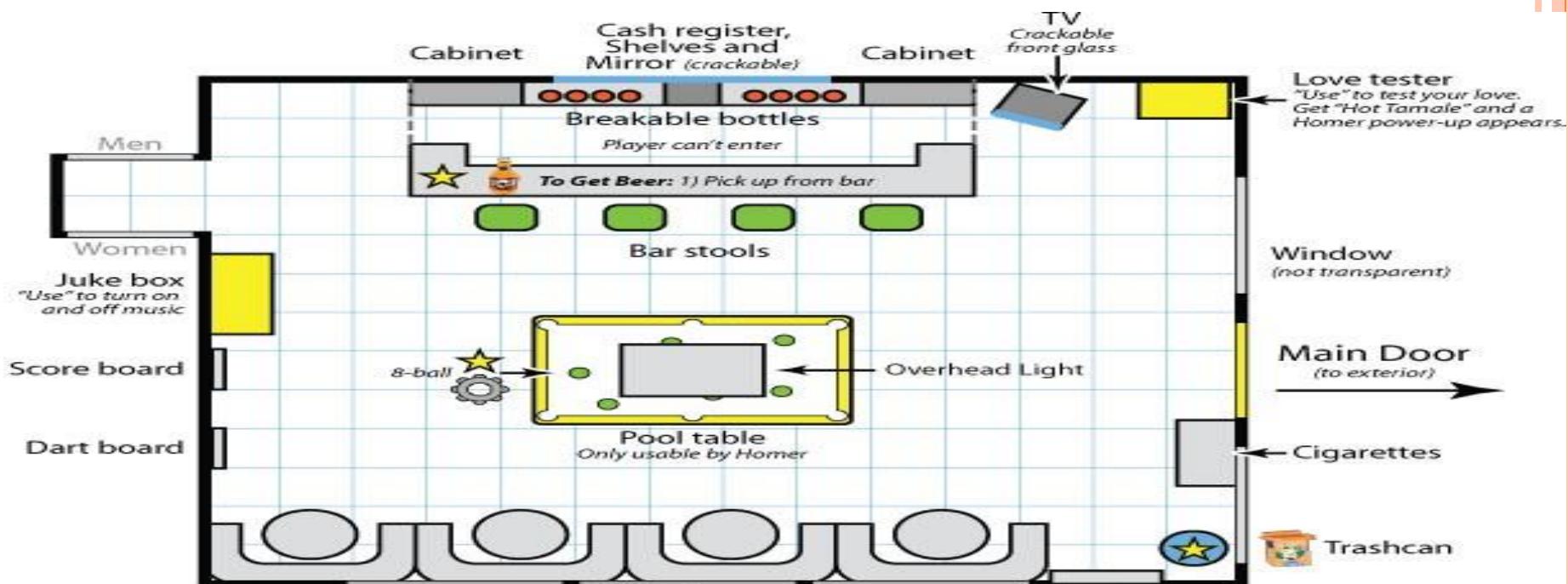


HorrorMage
Range fireballs







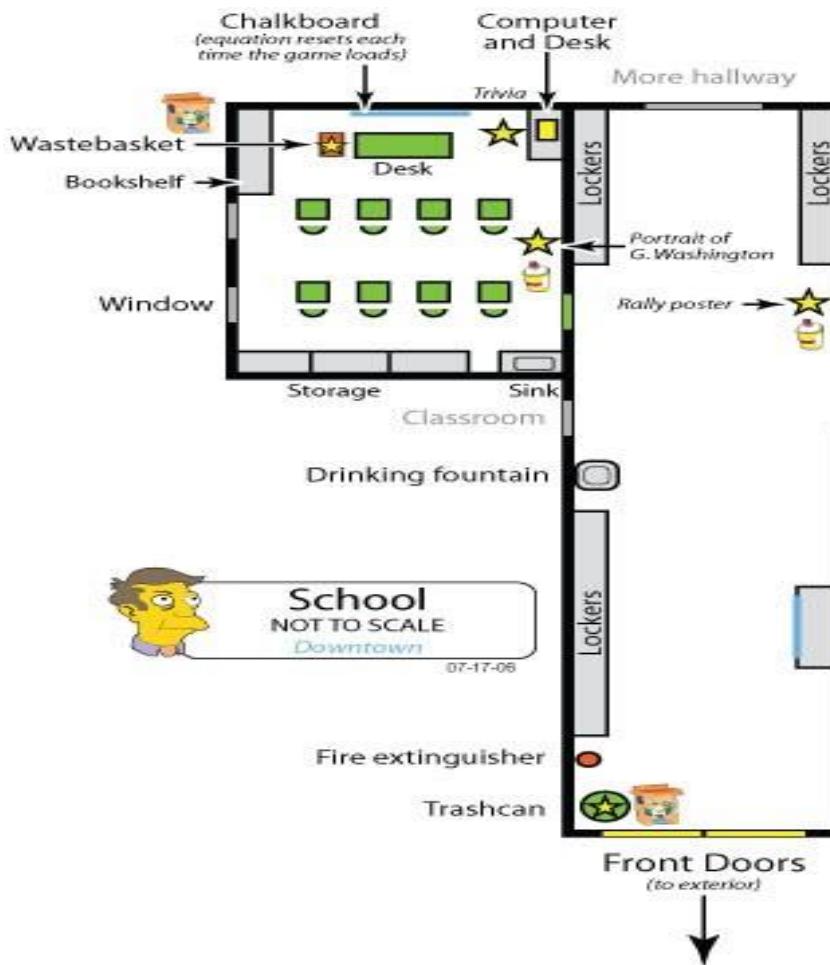


Moe's Tavern

TO SCALE

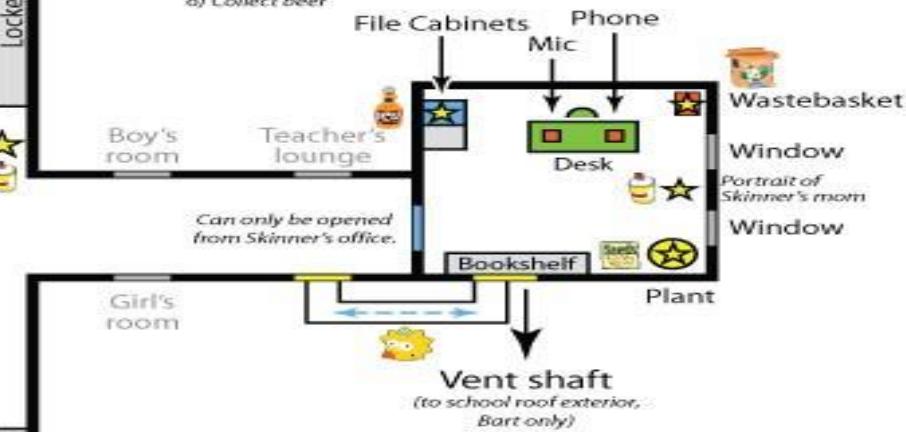
Downtown

07-06-06



To Get Beer:

- 1) Bart goes through roof vent
- 2) Bart opens Skinner's door
- 3) Homer enters Skinner's office
- 4) Homer hits or bumps file cabinet
- 5) Cabinet drawer opens
- 6) Collect beer





DESIGN DOCUMENTS

1. Game Mechanics

1. Core Gameplay
2. Game Flow
3. Characters/Units
4. Gameplay Elements
5. Game Physics
6. Statistics
7. AI
8. Multiplayer

2. User Interface

1. Flow chart
2. Functional Requirements
3. Mock-up
4. Buttons, icons, pointers

3. Art and Video

1. Goals, style, mood
2. 2D art and animation
 1. GUI
 2. Special Effects
3. 3D art and animation
4. Cinematics

4. Sound and Music

1. Goals, style, format
2. Sound effects
 1. GUI
 2. stceffe laicepS.4.2.2
 3. tnemnorivnE.4.2.3
3. Music
 1. Events
 2. System screens
 3. Level theme
 4. Situations
 5. Cinematic soundtrack

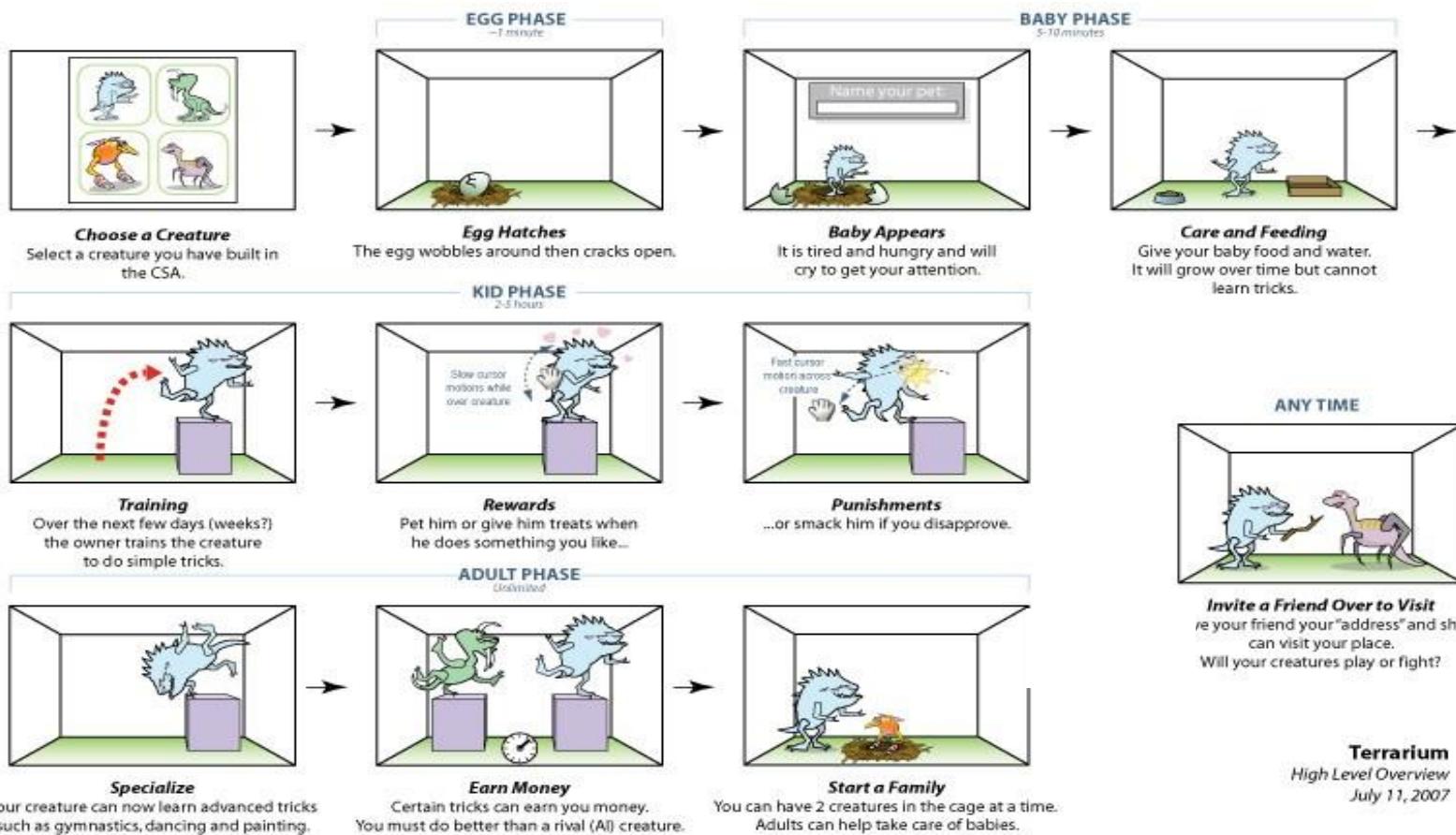
5. Story

1. Backstory and world
2. Character descriptions
3. Game text, dialog requirements
4. Sample scripts

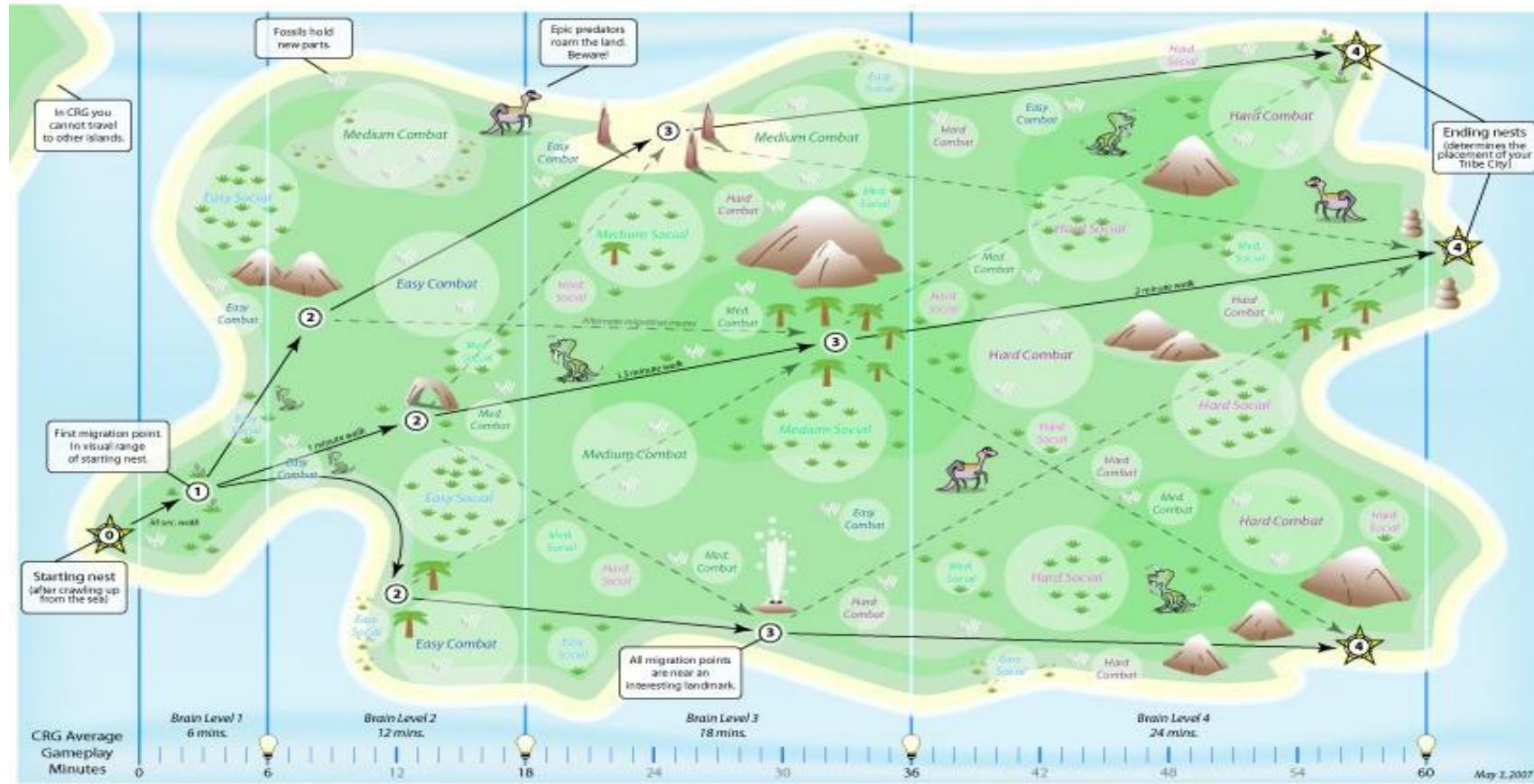
6. Level Requirements

1. Level Diagrams
 1. Flow diagrams
2. Asset revelation schedule

STORYBOARDS



TIME + SPACE



Title

date

Lots of whitespace!

Callout

Sidebar

- Bullet points
-
-
-

Main Illustration

Description

Detail Illustration

Notes

Callout



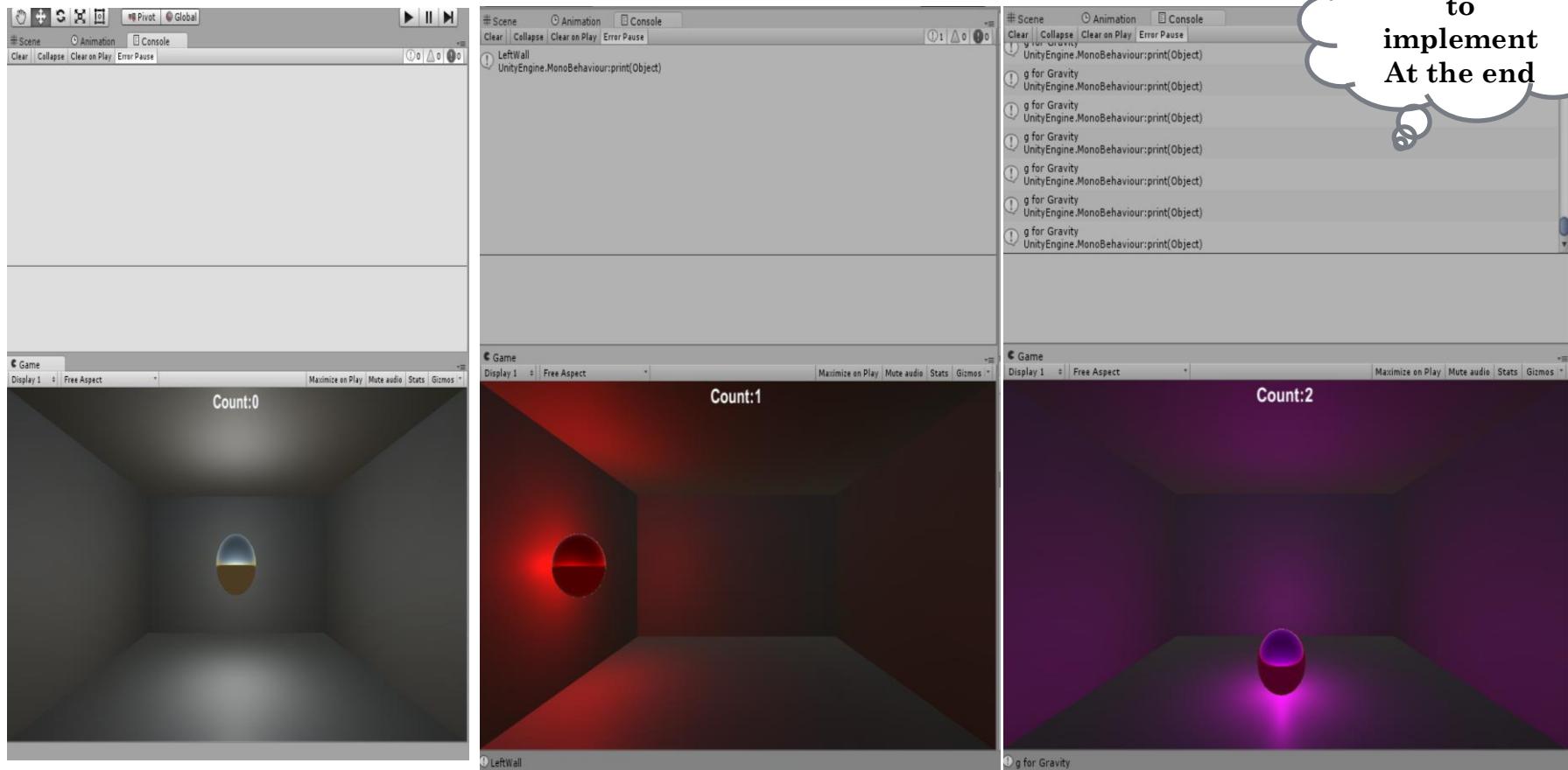
BUT SIMPLY ...

- • Don't over-concept and over-document
- • Prototype early
- • Prove features before entering production
- • Work iteratively as a group



ASSIGNMENT

Assignment
to
implement
At the end



DISCUSSION FILE

- Create a Test file for your game
- Create your game document
- **Date of discussion**