

# SE2250 – Software Construction

---

## Lecture #2

### Integrated Development Environment

January 14, 2019

---

■ **Tentative** schedule (sections 002/003):

- Assignment 1: Jan 29/28
- Assignment 2: Feb 12/11
- Project phase 1: Feb 26/25
- Midterm: Mar 11
- Assignment 3: Mar 19/18
- Project phase 2: Mar 26/25
- Project phase 3: Apr 9/8

# Outline

---

- Integrated Development Environments
- Unity
  - Panes
  - Project organization
  - Components
  - Understanding Colliders and Rigidbody
  - Prefabs
  - Watch for

# Integrated Development Environment

- Integrated Development Environments (IDE)
  - IDE is a software application that provides comprehensive facilities for software development.
  - Designed to maximize programmer productivity by providing components with similar user interfaces
- Normally include:
  - a source code editor
  - a compiler and/or an interpreter
  - build automation tools
  - a debugger

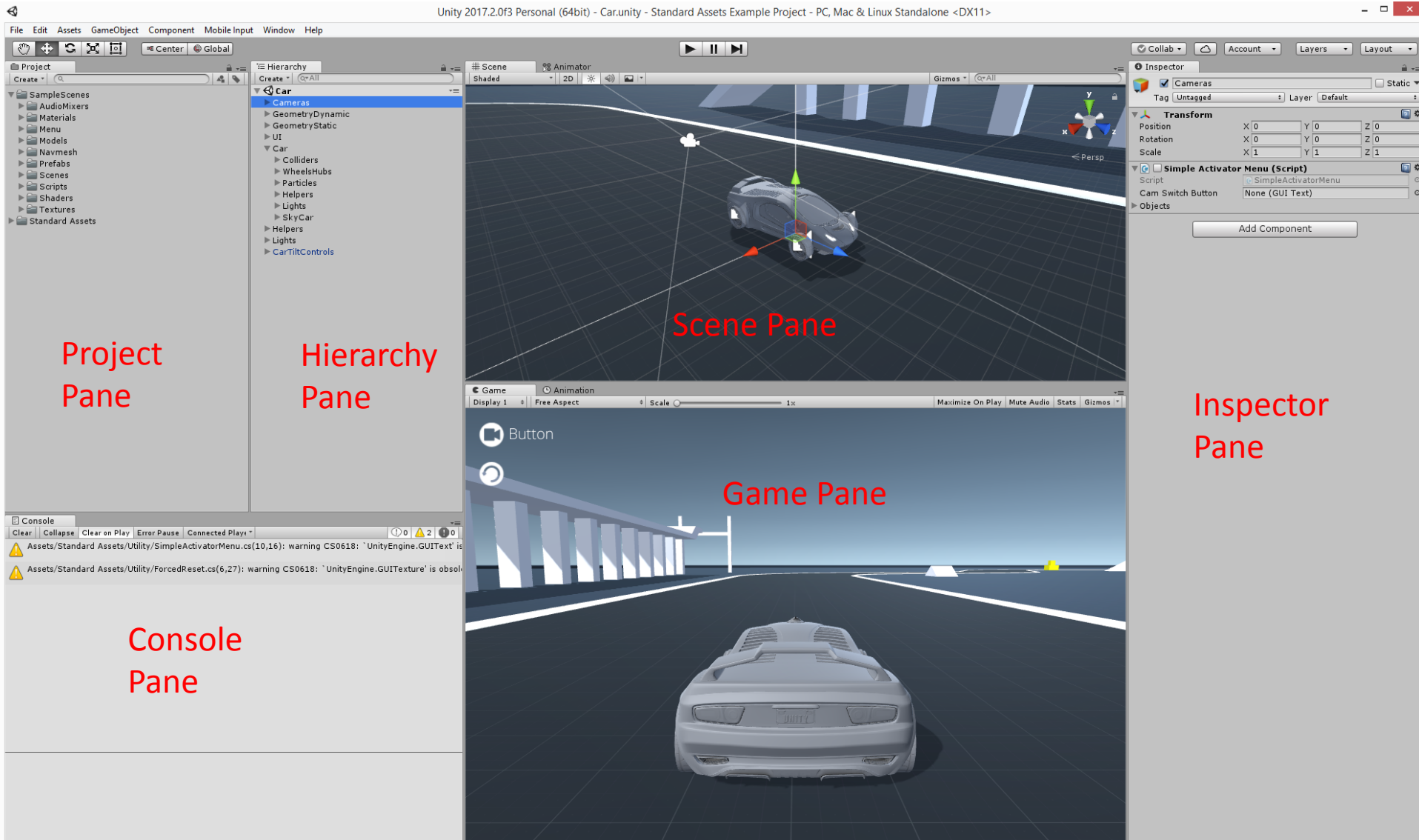
# Integrated Development Environment

- Examples: Visual Studio, Eclipse
- Some IDEs support multiple languages
- Many IDEs also have a class browser, an object inspector, and a class hierarchy diagram
- Sometimes a version control system and various tools are integrated

# Integrated Development Environment

- **SE2250 IDE → Unity**
- **MonoDevelop → Unity scripting engine**
- Game is contained in a project
- UI has windows/panes for different purposes

# Unity - panes



# Unity - panes

---

- **Scene pane:**

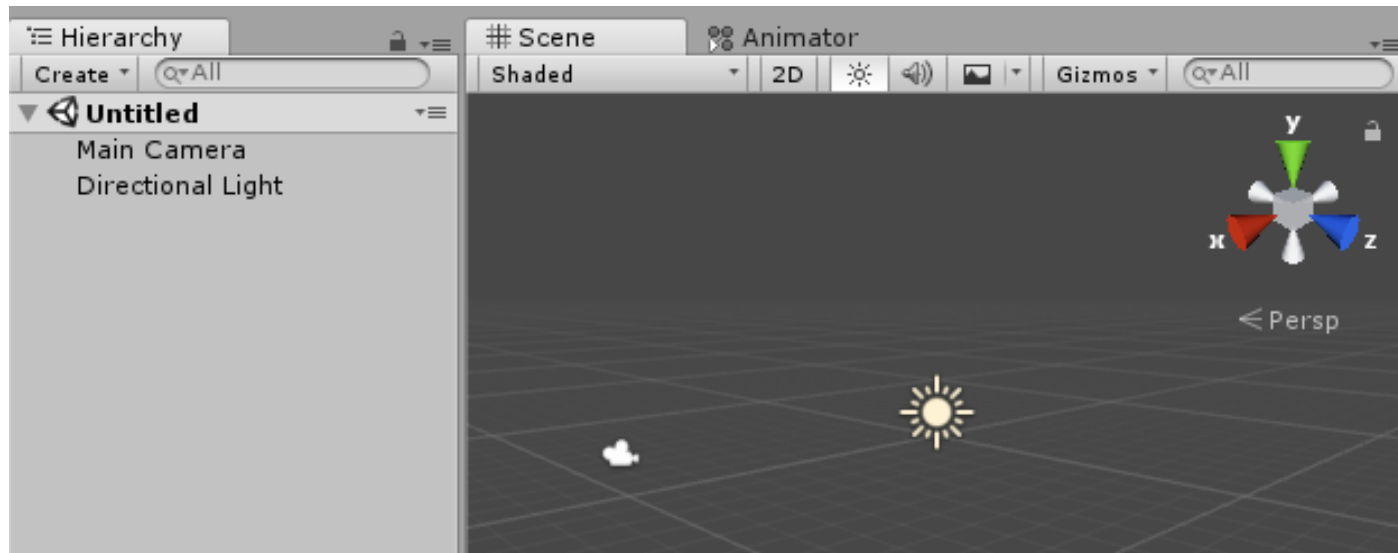
- Navigate around the scene in 3D
- Select, move, rotate, and scale objects
- You place environments, obstacles, decorations...
- Shows **scene** currently selected in the *project pane*
- When you create a new scene, it only has a Camera and a Light
- Tip: save the scene as soon as you create a project



# Unity - panes

---

- **Scene pane:**



# Unity - panes

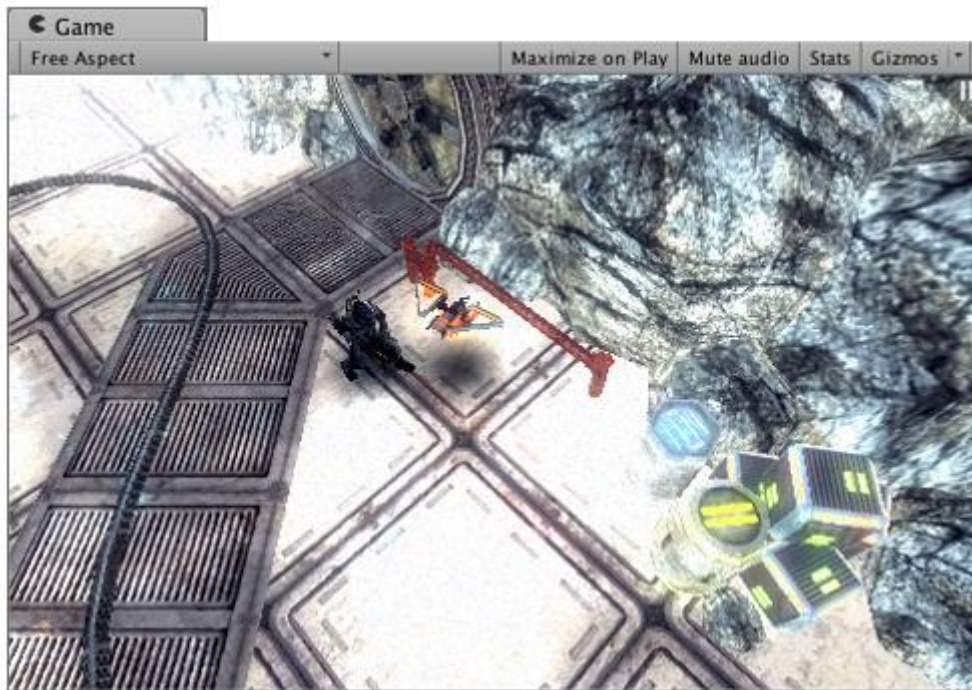
---

- **Game pane:**

- preview your actual gameplay
- the view from the Main Camera in your scene
- representative of your final, published game
- you will need to use one or more Cameras to control what the player actually sees

# Unity - panes

- **Game pane:**



- the buttons in the Toolbar control the Editor Play Mode



# Unity - panes

---

## ■ Hierarchy pane:

- Shows every GameObject included in the current scene
- Scene can represent game levels
- Everything in the scene is a GameObject (including camera)
- Can be instances of asset files (like 3D models) or prefabs

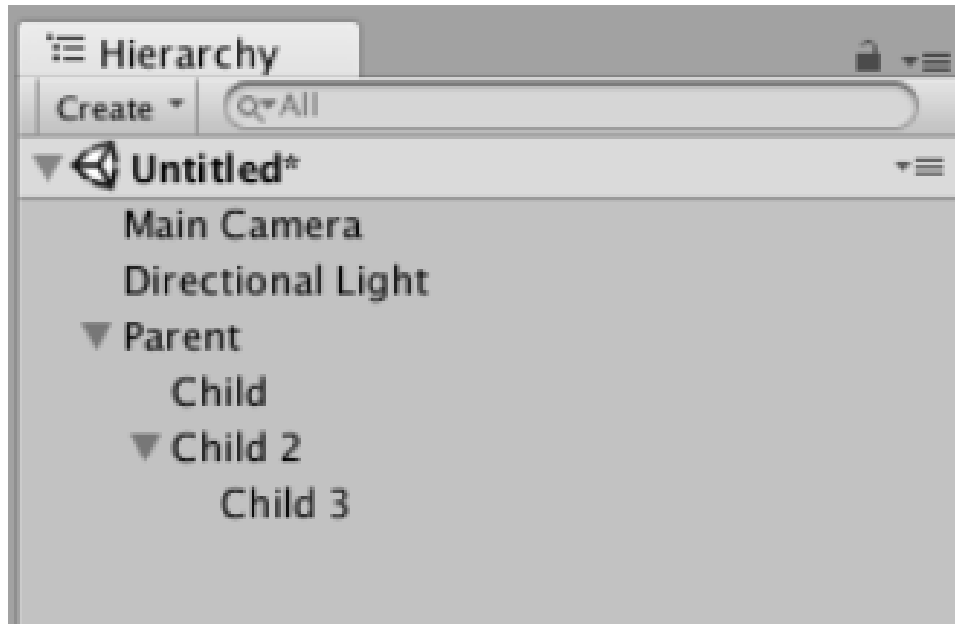


# Unity - panes

---

## ■ Hierarchy pane:

- During gameplay, as objects are added/ removed from the Scene, they will appear/disappear from the hierarchy.
- Parenting – grouping of objects (not like class/superclass)



# Unity - panes

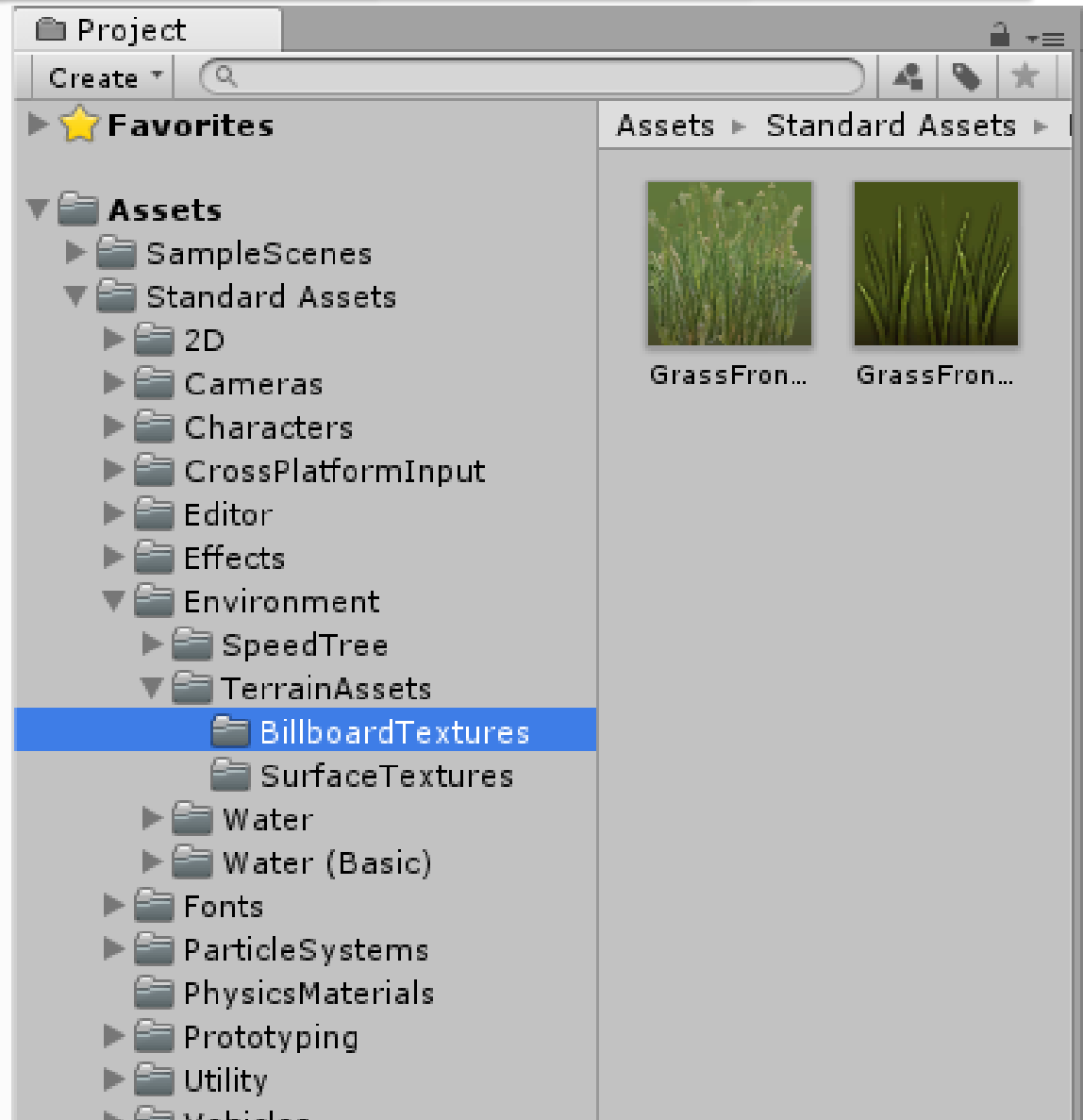
---

- **Project pane:**

- Contains all the assets that are part of the project (any kind of file that is part of the project: images, 3D models, C# code, text files, sounds, fonts etc.)
- Hierarchical display
- A reflection of the contents of the Assets folder within the Unity project folder
- Assets are not necessarily in the current scene.
- If using two column view, the right one shows the contents of the selected folder

# Unity - panes

- **Project pane:**

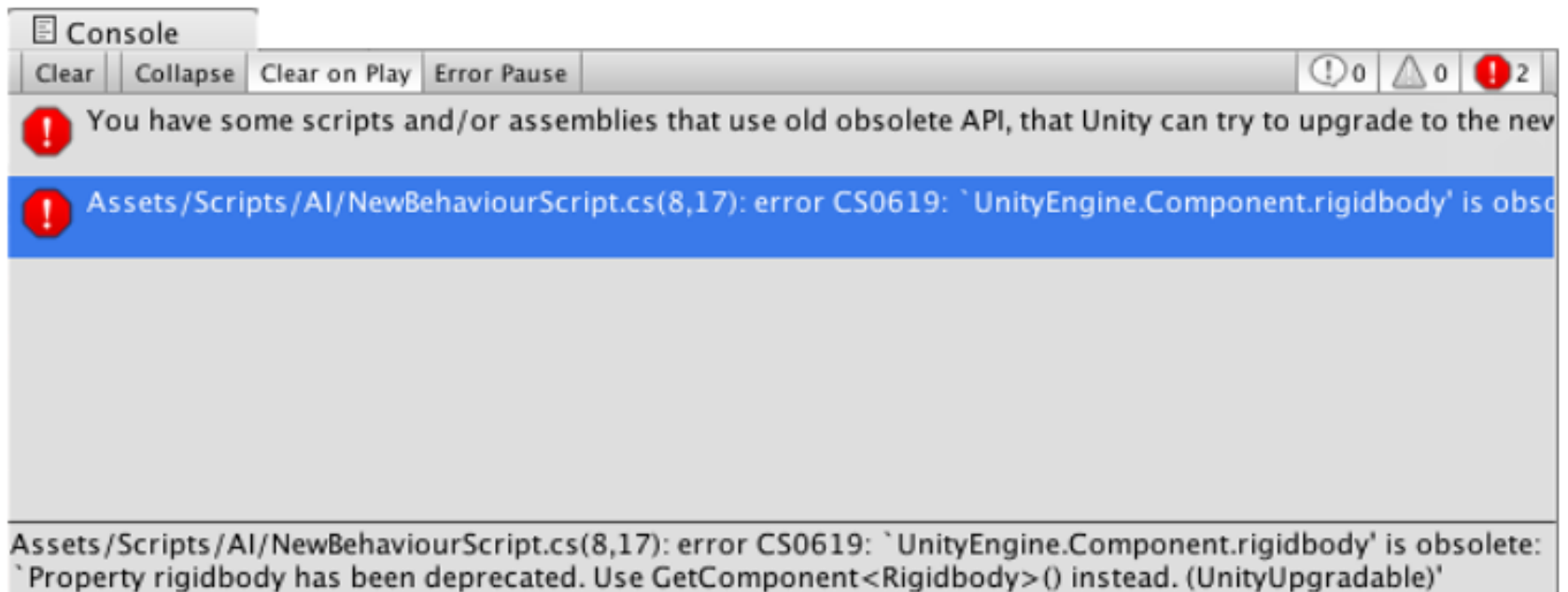


# Unity - panes

---

## ■ Console pane:

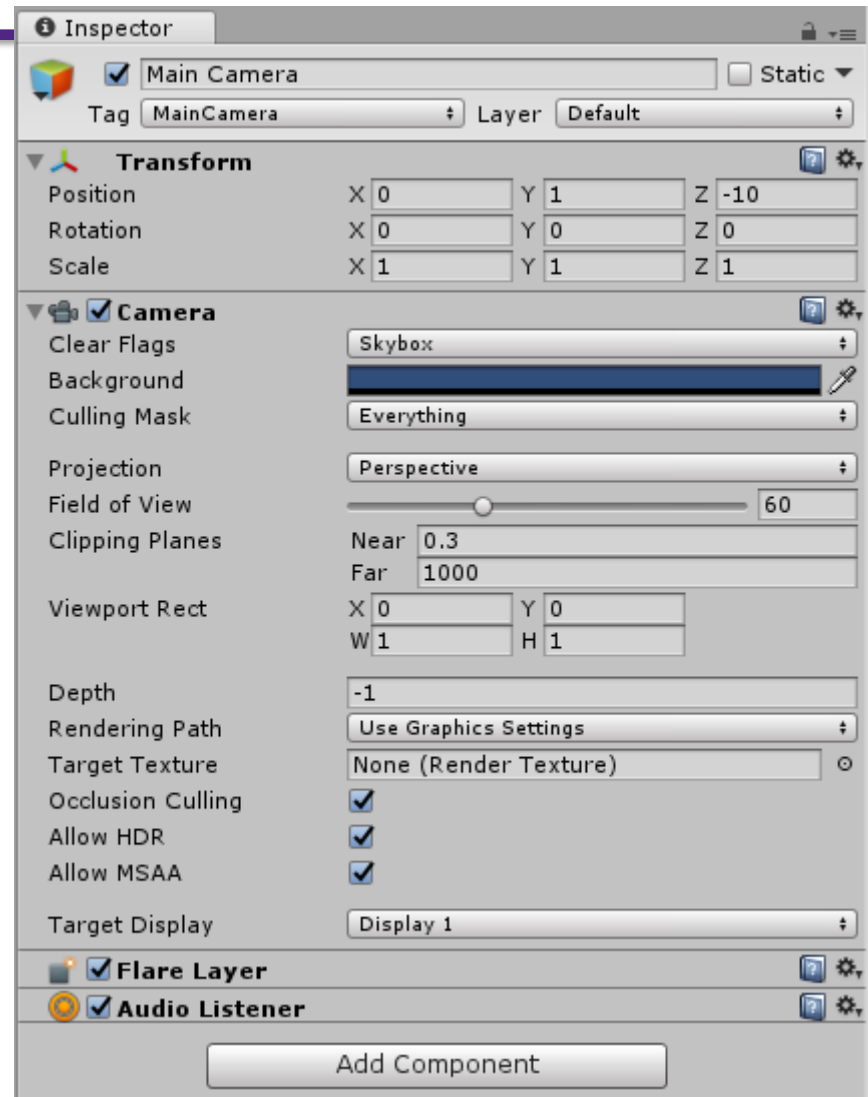
- Messages from Unity about errors or warnings
- Messages from yourself that will help you understand the inner workings of your own code





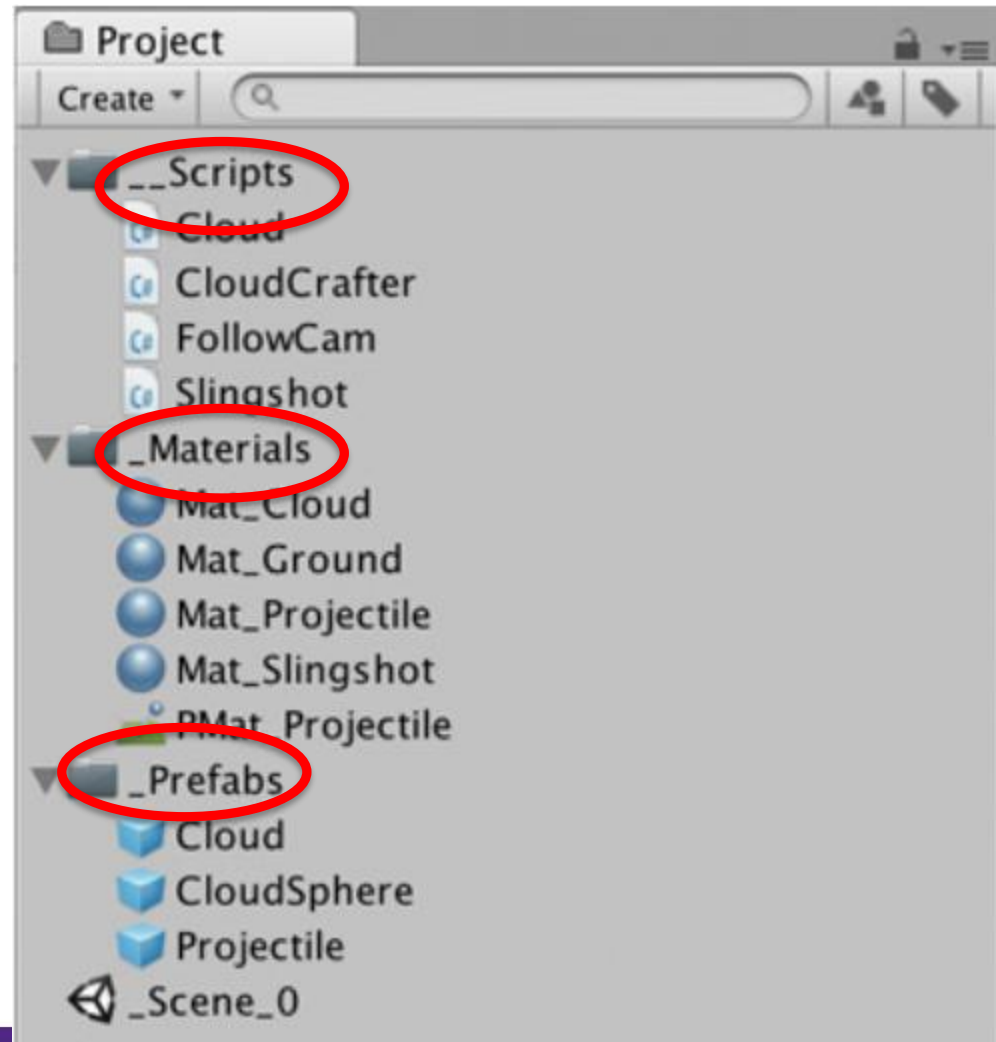
# Unity - panes

- **Inspector pane:**
  - Display and edit information about the currently selected asset in the Project pane or a GameObject in the Scene or Hierarchy panes.
  - Almost everything can be edited here



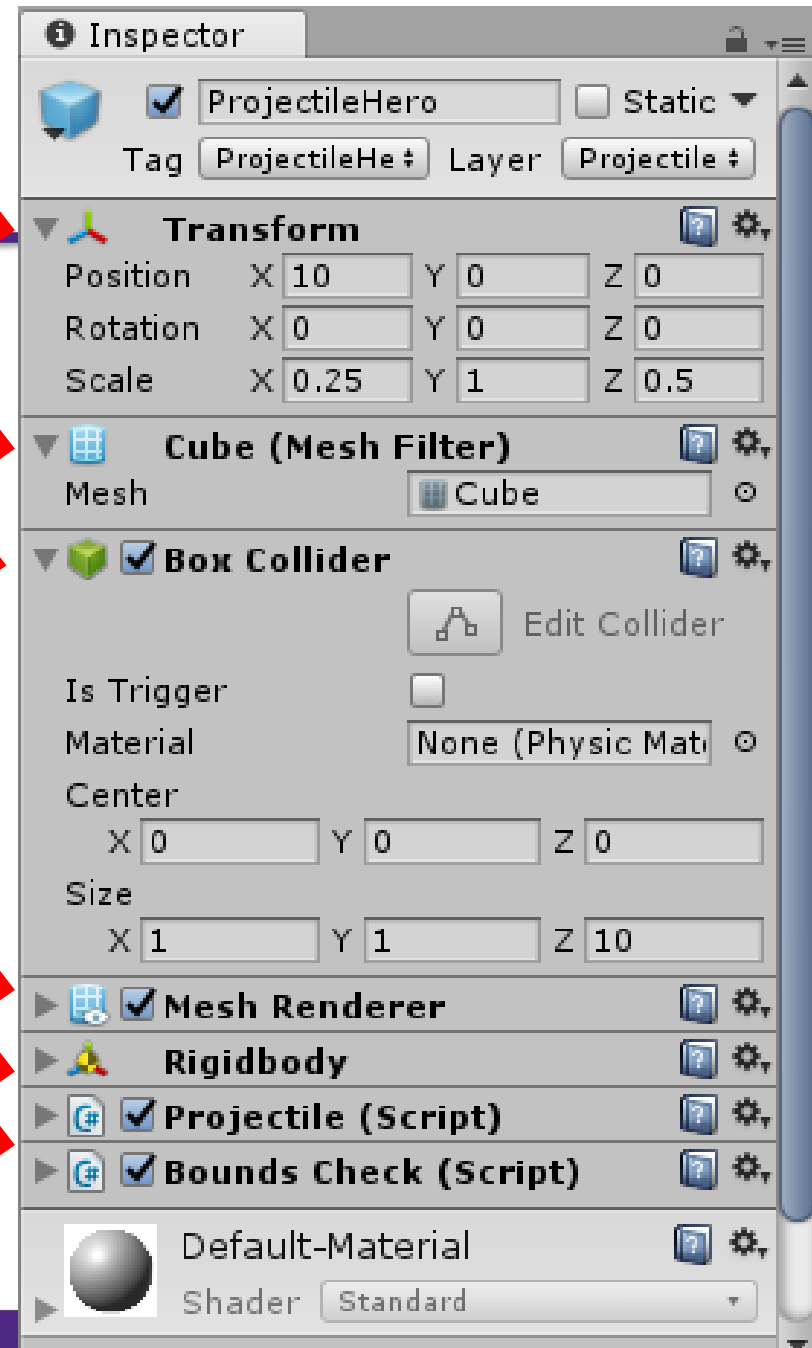
# Unity – project organization

- Organizing the project pane



# Unity – components

- Components
- Most Game Objects have components
- Components can be added and removed



# Unity – components

---

- Transform:
  - sets the position, rotation, and scale
  - the only required component
- Cube (Mesh Filter):
  - gives the GameObject its three-dimensional shape
- Mesh Renderer
  - makes that geometry visible (how it looks on the screen)
- Script(s)
  - how the objects behave

# Unity – components

---

- **Collider** (Box Collider in figure)
  - A collider enables a GameObject to interact with other objects in the physics simulation
  - Invisible
  - A rough approximation is often more efficient
  - Primitive colliders: box, sphere, capsule
  - Compound colliders – several colliders added to a single GameObject
  - Physics materials configure some physics: friction, bounciness...

# Unity – components

---

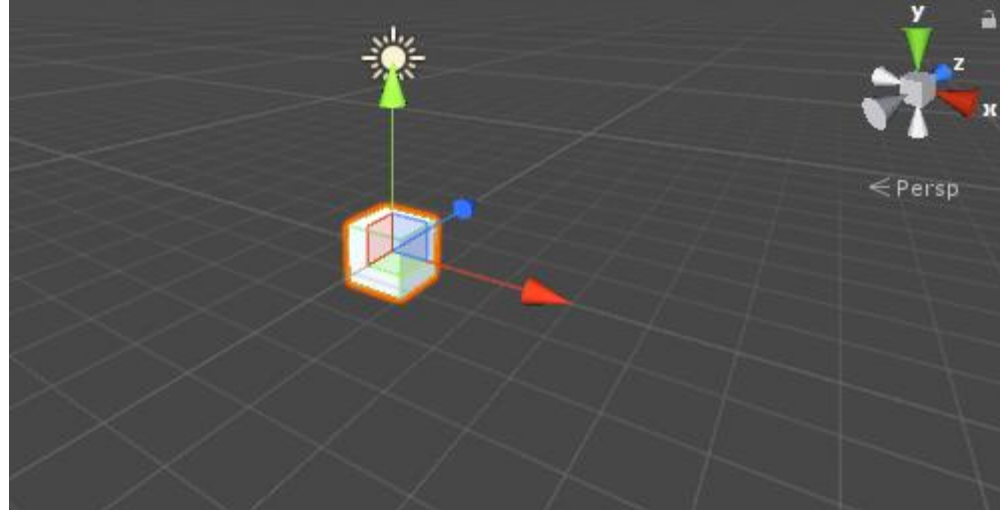
## ■ **Rigidbody:**

- Tells Unity that you want physics to be simulated for this GameObject
- Includes physical forces like gravity, friction, collisions, and drag.
- Without a Rigidbody, the Collider component of the GameObject will not move reliably
- Attach a Rigidbody component to any GameObject that you want to both move according to physics and properly collide with other colliders.
- Game object with a Rigidbody component - move it applying forces

# Unity - example

---

- Add a cube (GameObject > 3D Object > Cube from the menu bar)
- Set position to (0,0,0)
- Click the play button
- What happens?
- Why?
- Add Rigidbody



# Unity - Understanding Colliders and Rigidbody

---

## Rigidbody

- A Rigidbody attached, the object will respond to gravity
- If Collider components are also added, the GameObject is moved by incoming collisions
- **Is Kinematic** - removes it from the control of the physics engine and allows it to be moved from a script.



# Unity - Understanding Colliders and Rigidbody

---

## Colliders

- Colliders without Rigidbody
  - static colliders
  - motionless elements - walls, floors
- Is trigger
  - detect when one collider enters the space of another without creating a collision
  - will simply allow other colliders to pass through
- Physics materials
  - simulate the properties of the material
  - configure friction and bounce

# Unity - Understanding Colliders and Rigidbody

---

## Interaction

- Static Collider
  - stays at the same place
  - should not be moved during play
- Rigidbody Collider (most common)
  - fully simulated by the physics engine
  - can react to collisions and forces applied from a script
  - can collide with other objects (including static colliders)
- Kinematic Rigidbody Collider
  - move from the script
  - do not respond to collisions and forces
  - will apply friction to other objects and will “wake up” other rigidbodies

# Unity - Understanding Colliders and Rigidbody

- Collision action matrix
  - Collision detection occurs and messages are sent upon collision

	Static Collider	Rigidbody Collider	Kinematic Rigidbody Collider
Static Collider		Y	
Rigidbody Collider	Y	Y	Y
Kinematic Rigidbody Collider		Y	

# Unity - Understanding Colliders and Rigidbody

---

- Move the cube from the script
  - In the Cube inspector select *Add Component>New Script*
  - Give script a name *Cube*, select *Create and Add*
  - Open the script
  - MonoDevelop opens – unity scripting editor

# Unity - Understanding Colliders and Rigidbody

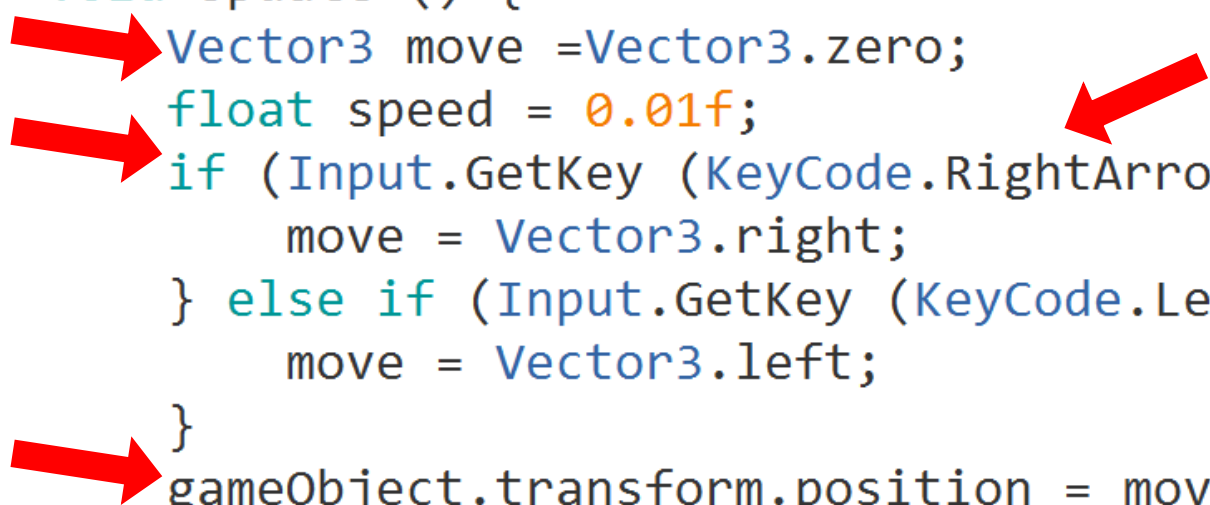
- Move the cube from the script

```
5 public class Cube : MonoBehaviour {  
6  
7     // Use this for initialization  
8     void Start () {  
9  
10    }  
11    |  
12    // Update is called once per frame  
13    void Update () {  
14  
15    }  
16 }  
17
```

# Unity - Understanding Colliders and Rigidbody

- Move the cube from the script

```
12  // Update is called once per frame
13  void Update () {
14      Vector3 move = Vector3.zero;
15      float speed = 0.01f;
16      if (Input.GetKey (KeyCode.RightArrow)) {
17          move = Vector3.right;
18      } else if (Input.GetKey (KeyCode.LeftArrow)) {
19          move = Vector3.left;
20      }
21      gameObject.transform.position = move*speed
22      + gameObject.transform.position;
23  }
24 }
25
```



# Unity - Understanding Colliders and Rigidbody

---

- Is the cube moving with arrow clicks?
- Create another cube (without the script attached)
- What happens now?

# Unity - Understanding Colliders and Rigidbody

---

- Create a cube from a script
  - Change the moving cube?
    - Is kinematic = on
    - What happens?
  - Change the static cube?
    - Is Kinematic = on
    - What happens?
- *Kinematic rigidbodies will not be affected by forces or collisions, but will collide with non-kinematic rigidbodies, sending collision events and pushing them out of the way*



# Unity - Prefabs

---

- Create cubes from script?
- Prefab
  - Allows you to store a GameObject object complete with components and properties.
  - A template from which you can create new object instances in the scene
  - Sometimes we create objects in the scene just to convert them into prefabs
  - Changes to a prefab reflected in instance, but can override and set for each instance

# Unity - Prefabs

---

- To create Prefab
  - Create a GameObject in the scene
  - Drag the object from the hierarchy pane to the project pane



- Note: Assets are not organized in this snapshot

# Unity - Prefabs

---

- To create cubes from the script:
  - Create Main script and attach it to the Main camera
  - Can remove the static cube from the scene
- Note: a script must be attached to an object to execute

# Unity - Prefabs

---

- To create cubes from the script:

```
public GameObject      cubePrefabVar;  
private GameObject     cube;  
// Use this for initialization  
void Start () {  
    Vector3 offset = new Vector3 (3, 0, 0);  
  
    for (int i=0;i<5; i++){  
        cube = Instantiate(cubePrefabVar);  
        cube.transform.position  
            = cube.transform.position + i*offset;  
    }  
}
```

# Unity - Prefabs

---

- To create cubes from the script:
- Will it run?
- What's missing?

# Unity – watch for

---

- 90% of bugs are just typos.
  - Misspellings
  - Capitalization
  - Missing semicolons
  - “==” and “=”
- Changes done while the game is running do not persist