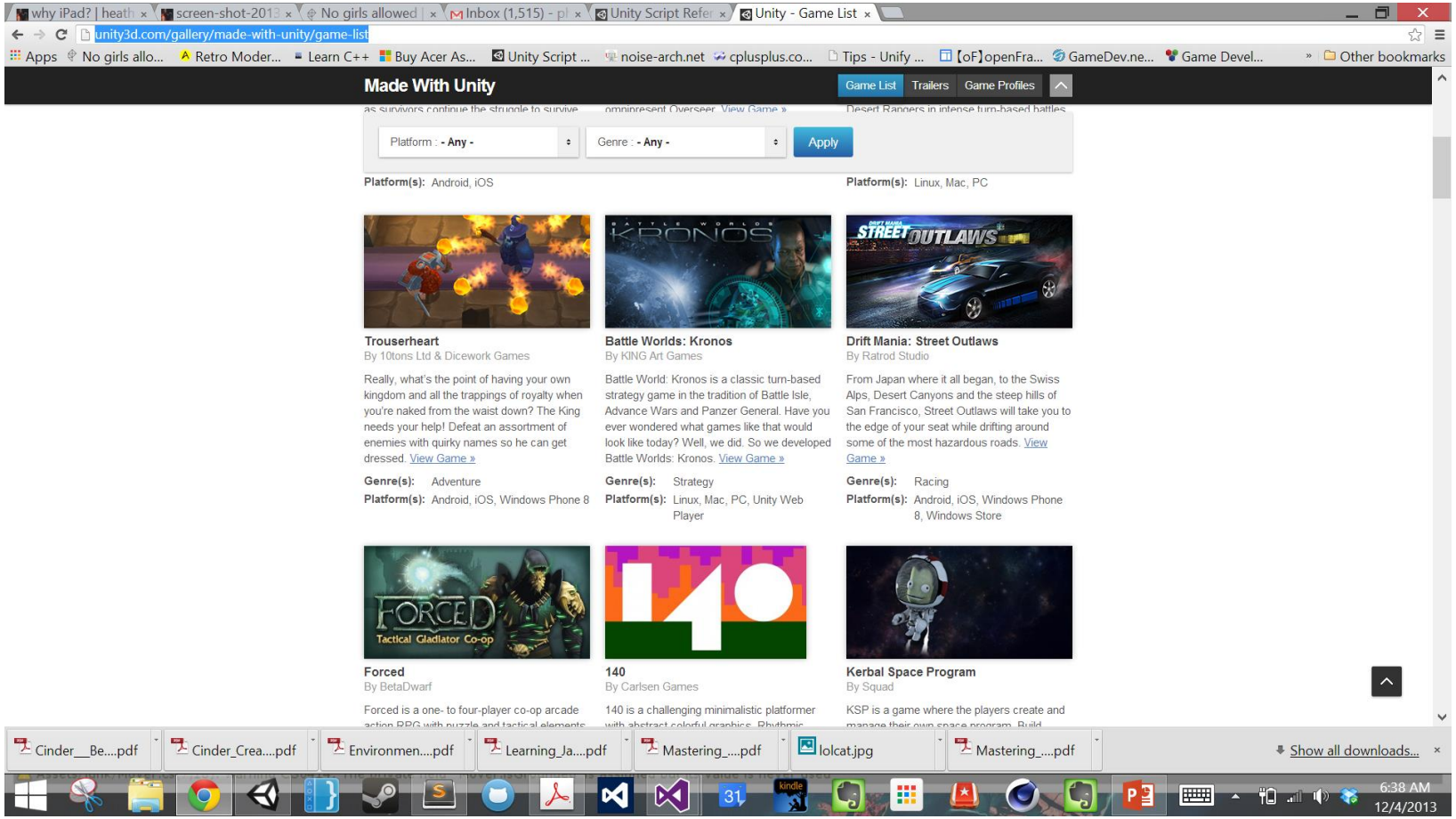
Unity

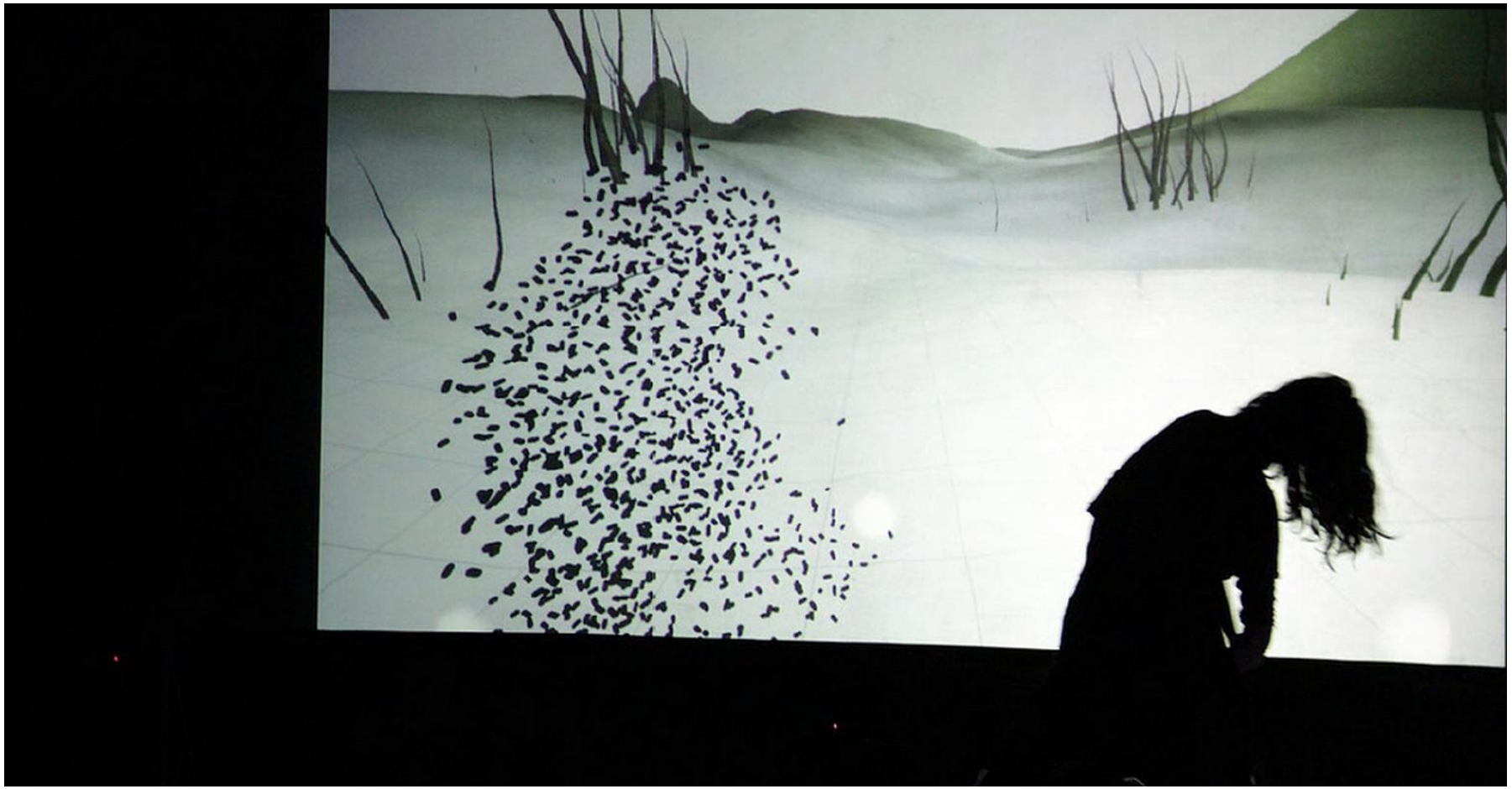
# Unity is a Game Engine.



# Unity comes with prebuilt functionality speeding development drastically

- Collision detection
- Key presses / input
- Animation engine
- Physics
- Lighting
- Cameras
- Terrain
- 2D + 3D worlds
- Publishing to multiple platforms
- OpenGL set up

# Unity is also used for installation



Swimming with Particles

http://www.creativeapplications.net/other/dancing-with-swarming-particles-kinect-unity/

# Good indie Unity games

- Device 6
- Gone Home
- 140 http://vimeo.com/59001919
- Mirror Moon http://www.youtube.com/watch?v=s2l3h4AeDXI
- The Silent Age

# The Unity Platform SDK

Best learning resources:

- http://unity3d.com/learn/documentation
- http://www.unity3dstudent.com/
- catlikecoding.com
- http://unitygems.com/
- Unity Forum - http://forum.unity3d.com/forum.php

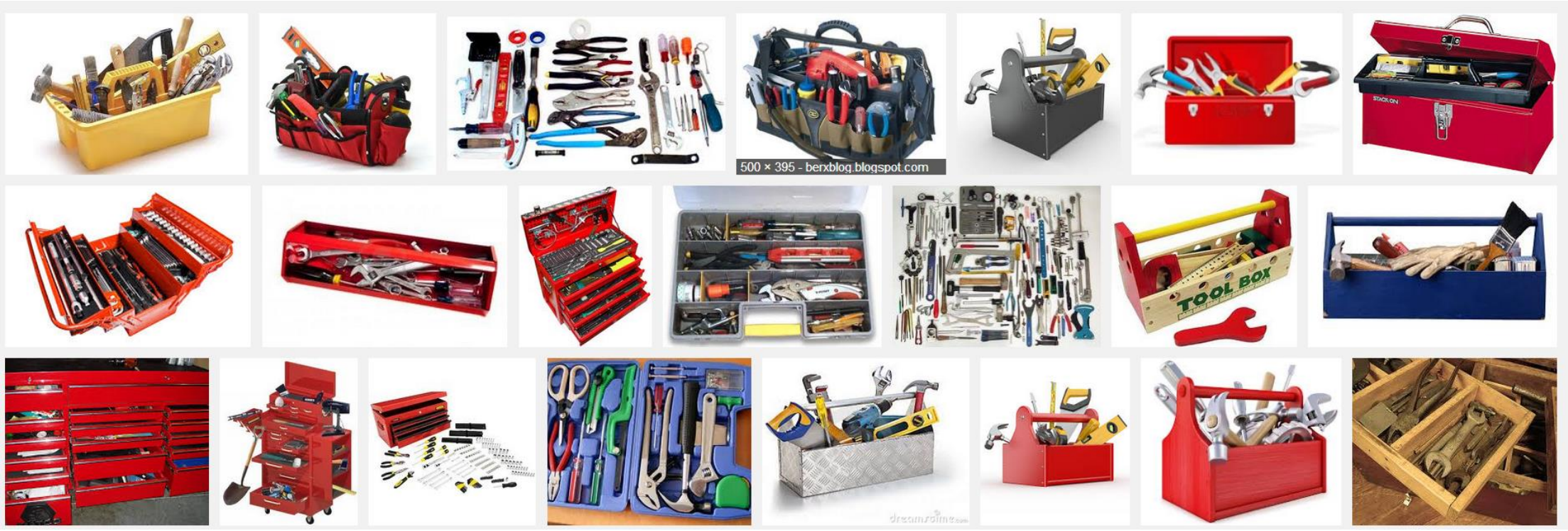
# Everything in Unity is a Game Object.

Think of game objects like empty tool boxes.

- They just are empty boxes waiting to be filled with tools, or COMPONENTS.
- Components do things. They are the tools in the box.
- All game objects have a Transform component that controls position, scale and rotation. This is like your hammer – every tool box has to have one.
- You can script all public properties of any component, allowing you to change things over time, such as animation, color, visibility, user input, on screen text and more

# In fact, you can think of unity game objects like tribbles – they just keep multiplying!

# There are many types of tools and not all tool boxes are equal even though they share many of the same components.



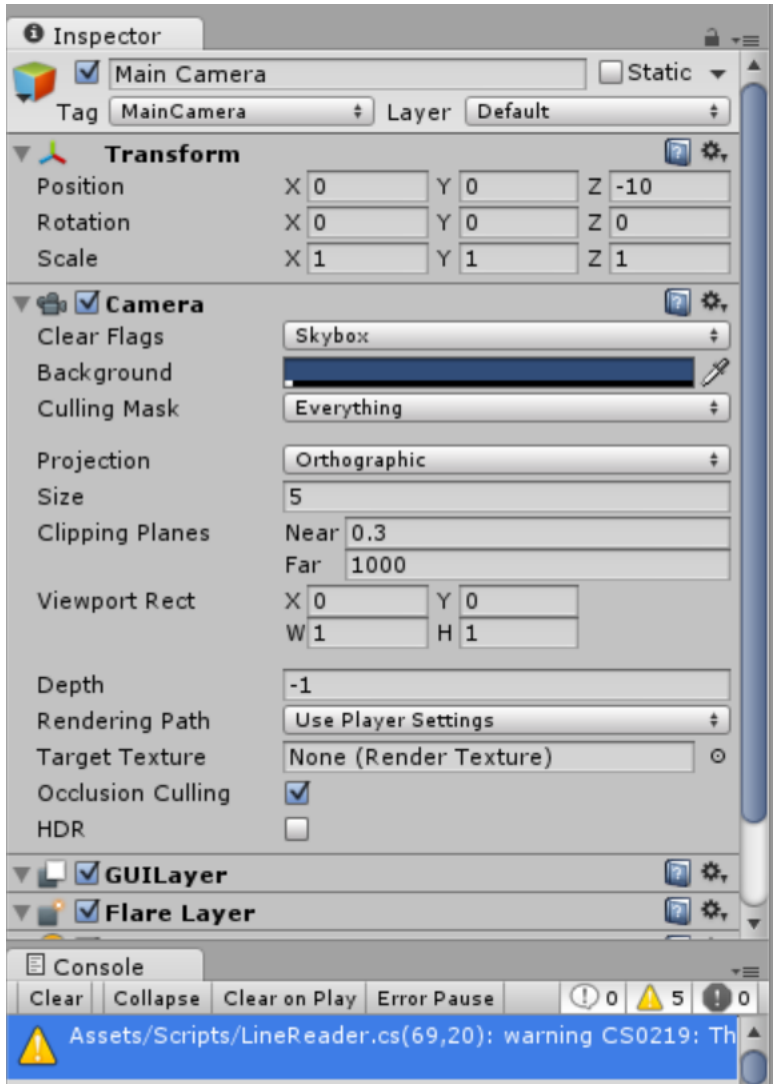
# Components do things.

Unity comes with some pre-set up Game Objects with all of the components needed for common tasks

- Particle System, cameras, Cubes, Spheres, Planes, Sprite (2d game object), lights
- All of these pre-setup objects live in the main menu under Game Object.
- There can also be Empty Game Objects in your game. These objects are extremely useful for adding scripts to that create things during the game or use multiple game objects at once.

# Unity's Inspector

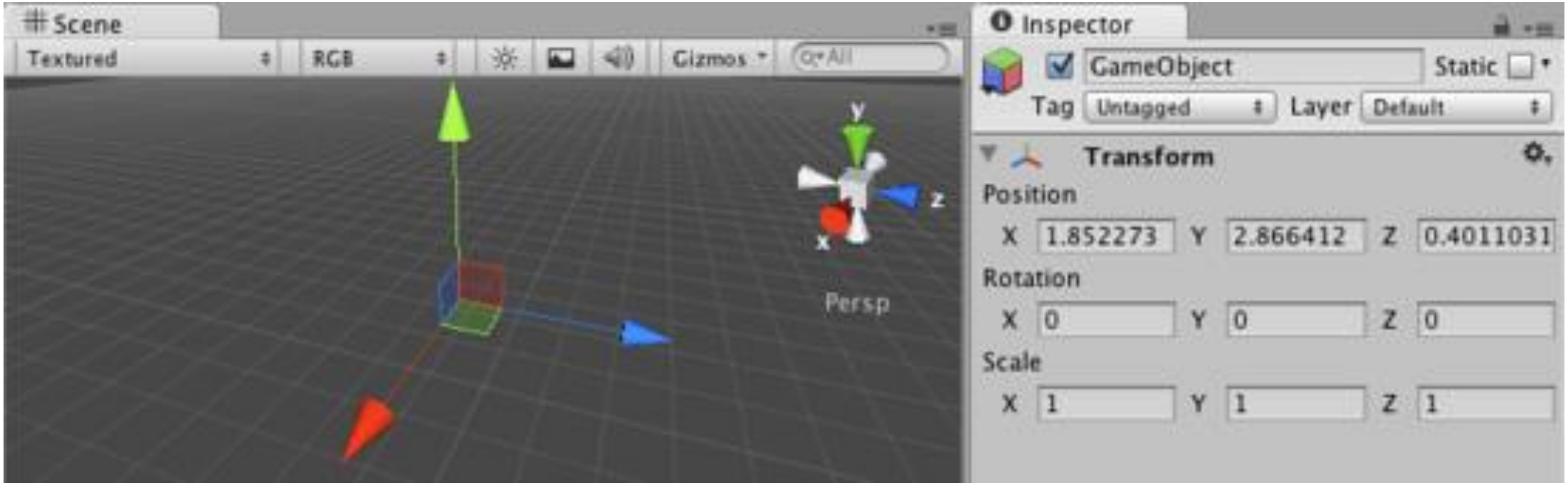
- This area lets you see what components are on a game object and add new components
- Makes visible all component public properties for editing



# Other tabs

- Hierarchy – all objects currently in your game world
- Project – your files in your project folder on your computer's hard drive. Not all objects in Project are even in your game. They are just things you might use or will use. It's best to keep this spot organized.
- Game view – What the camera in the game is looking at
- Scene view – A freely positional view that has no connection to the active camera or what the player sees – good for lay out and design.
- Other views exist under Window and are ctrl 0 -9 on the keyboard

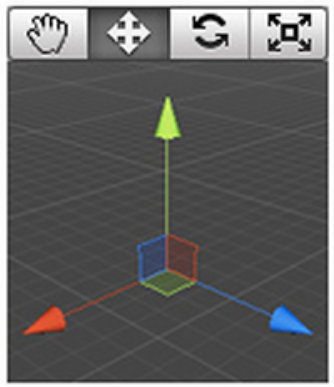
# The Transform gizmo or x, y, z



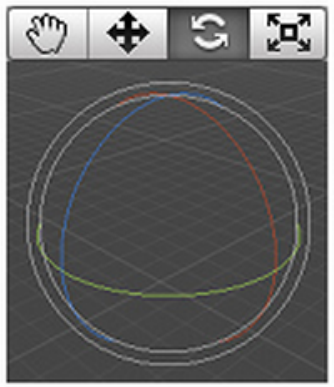
# Transform tools

*The View, Translate, Rotate, and Scale tools*

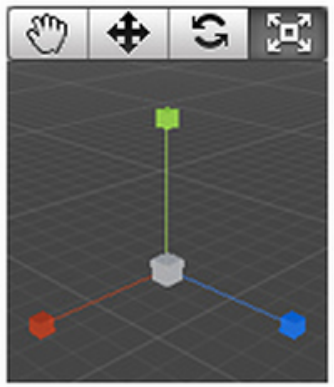
The tools can be used on any object in the scene. When you click on an object, you will see the tool gizmo appear within it. The appearance of the gizmo depends on which tool is selected.



Translate (W)



Rotate (E)



Scale (R)

All three Gizmos can be directly edited in the Scene View.

When you click and drag on one of the three gizmo axes, you will notice that its color changes. As you drag the mouse, you will see the object translate, rotate, or scale along the selected axis. When you release the mouse button, the axis remains selected.

# Scenes

- Levels are scenes in Unity.
- You can make as many as you like
- You can link them up to play one after another in the build settings
- When you save your scene you can see it in the assets folder and double click on it launch it

# Mono Develop

Mono Develop is Unity’s default code environment

Creating a script in Unity will allow you to open it. Lets do that now and start making some basic code.

The console is Unity’s debug log. We will also be looking at that.

# The console is an amazing window

- Go to the main menu > Window > Console

This lets you print out stuff to screen by using the simple command

Debug.Log(“I love indiegames”);

Exmaple: aSimpleScene

# C#

Tonight we will be learning a bit of c#.

C# is an amazingly powerful language. All of windows runs off it. Journey was made using it, as were many, many of the games you have ever played.

It is Unity’s most powerful language

# What are types & statements?

You can think a type as *the kind of thing* something is.

7 ***is a*** number

# Useful primitive types

These types are commonly referred to as primitive types and the are included in the c# language.

- int = is a whole number
- float = is a decimal
- string = is a word
- bool = is true or false value

# Complex Unity types

Unity has some built in types for in game friendly behavior

- GameObject is an object in your game
- Collision2D is a shape around your graphic that lets you know if it hits something else in your game
- Rigidbody2D is a type that makes your object respond to physics

# Hello semicolon

# Statements are lines of code followed by a semicolon

int x = 5;
float y = 9.0f;

(in c# you need an f after the last number in a float or decimal number)

The = sign is the assignment operator, and it works just like it does in algebra. What's on the right becomes equal to what is on the left.

**Variables** have a type and they can be changed at any point in time. They vary....

```
int x = 5;
x=6;
```

Variables can change whenever you need them to! This is really great for things like game score

# You can think of a variable like a box holding data.

Is a Cardboard box holding baby kittens. You'd like to see those kittens right????

Example Scenes: types

# Scope Operators

```
{
    everything inside these braces is
    grouped together.
}
```

- Scopes can be nested.
- All variables inside a scope are visible to any scope inside of it.
- Any variable declared inside a scope ends when the scope ends

# Comments! MAKE THEM!

To make comments just put two slashed in front of your statement

Single line
```
//this variable keeps track of score
Int myScore = 10;
```

Multiline
```
/*
 use all the space
You might need
*/
```


*le coding*
Public Class Employee
Private _name As St
Private _salary As
Public ReadOnly Pro
Get
Return _nam
End Get
End Property
Public ReadOnly Pro
Get
Return _sal
End Get
End Property
Always put enough comments in your code!
Opening file 6 weeks later...
WHAT THE FUCK

# Good nested scopes

```
{
        int x = 6;
        {
                int y = 10;
        //we can see x here because this scope is enclosed within it. The
        corresponding scope brace is still open at this point.
                x = y;
        }
}
```

# Bad nested scopes

```
{
        int x = 10;

        {
                int y = 10;
        }

        //since y was declared inside the above scope, it's deleted when it's
        associated scope brace is seen. It no longer exists. This would break.
        y = x;

}
```

# Conditionals

Sets of instructions that run if something is true

```
int health = 1;
bool hasTreasure = true;

if(hasTreasure)
{
        heatlh = health +1;

}
```

Example:playerHealth

# You can also do simple evaluations in the parentheses

```
 int  score=10;


 if(score< 1)
{
        health = health -1;
}
```

# If this is true do this else do that.

```
if(time > 12)
{
        weeksGrounded = weeksGrounded + 2;
}
else
{
        raiseAllowance++;
        //short hand for adding 1 to raiseAllowance
}
```

# && and || or operators. Gang up conditions

```
if(heath > 10 && playerIsAlive)
        {
                do stuff
        }
```

**The && is and**
**You can use || for or as well**

```
If(health  > 10 || playerHasTreasure)
        {
                do stuff
        }
```

# Functions

- Functions are blocks of instructions –
- They have a few pieces
- Who can use it, what it returns, its name and optional parameters. We aren't covering parameters just now.

```
public void Instructions( parameters can optionally live here){
        doSomething();
}

private int Instructions ()
{
        int x = 5+6;
        return x;
}
```

# Functions are machines. They simply do things

# Function calls

When you want to run these instructions you call them in your code

```
makeCoffee();
```

# Function body

This is the function definition and the body

```
public void makeCoffee()
{
        Debug.Log(“making coffee”);
}
```

# Dot Syntax

You can get access to a Game Object's or component's functions and variables using the dot syntax

```
GuiText myText = guiText.text;
Ship.Instructions();
Ship.x = 6;
```

In this class we will be using many unity built in function. For more info see the Unity site and SDK

# Component can be code

Scripts are components in Unity.

You define them yourself and attach them to the default Unity Game Objects.

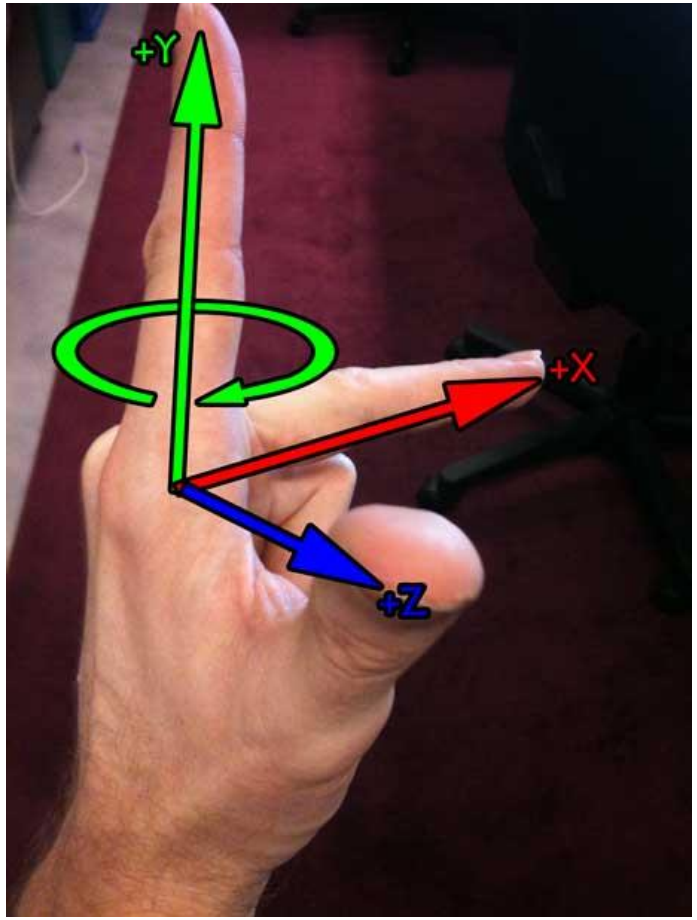
Your Scripts contain all of the functions and variables you might need for your game.

# Vectors!

Points in either 2D 3D space contained one object.

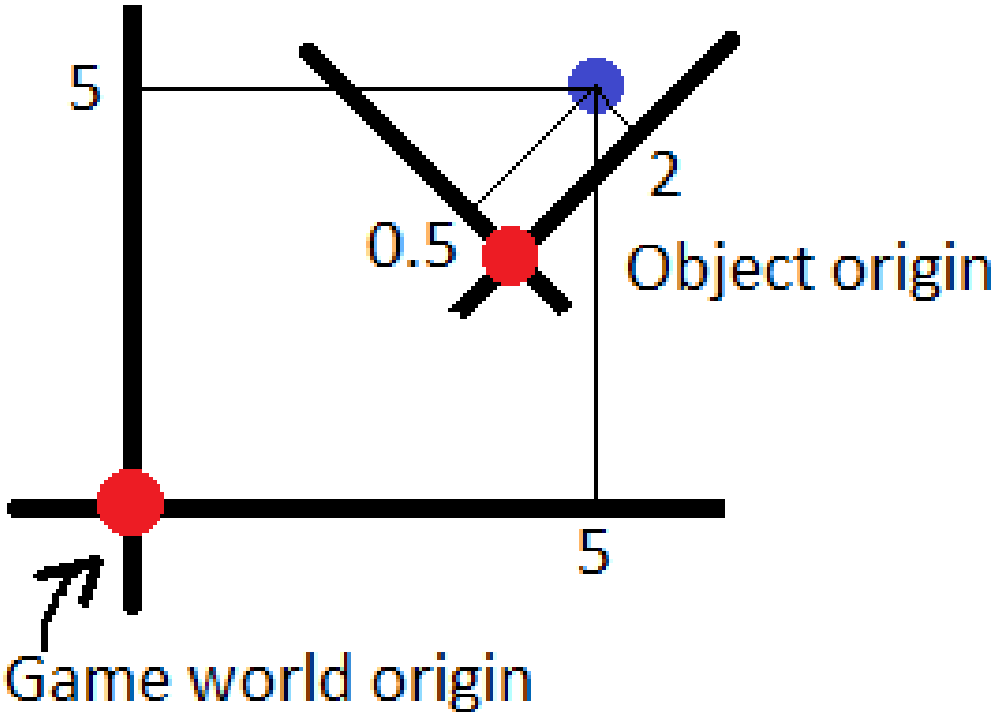
To create
Vector3 myVect = new Vector3(1.0f, 0.0f, 1.0f);
Vector2 myVect2 = new Vector2(10.2f, 10.1f);
myVect.x = 10.0f;
myVect.z = 1.0f;
Position, rotation and scale  in unity is stored as a Vector3 variables
transform.position;
To get access to the x, y, or z positions separately? Use dot syntax
transform.position.x
tranform.rotation.x

# Unity is left handed

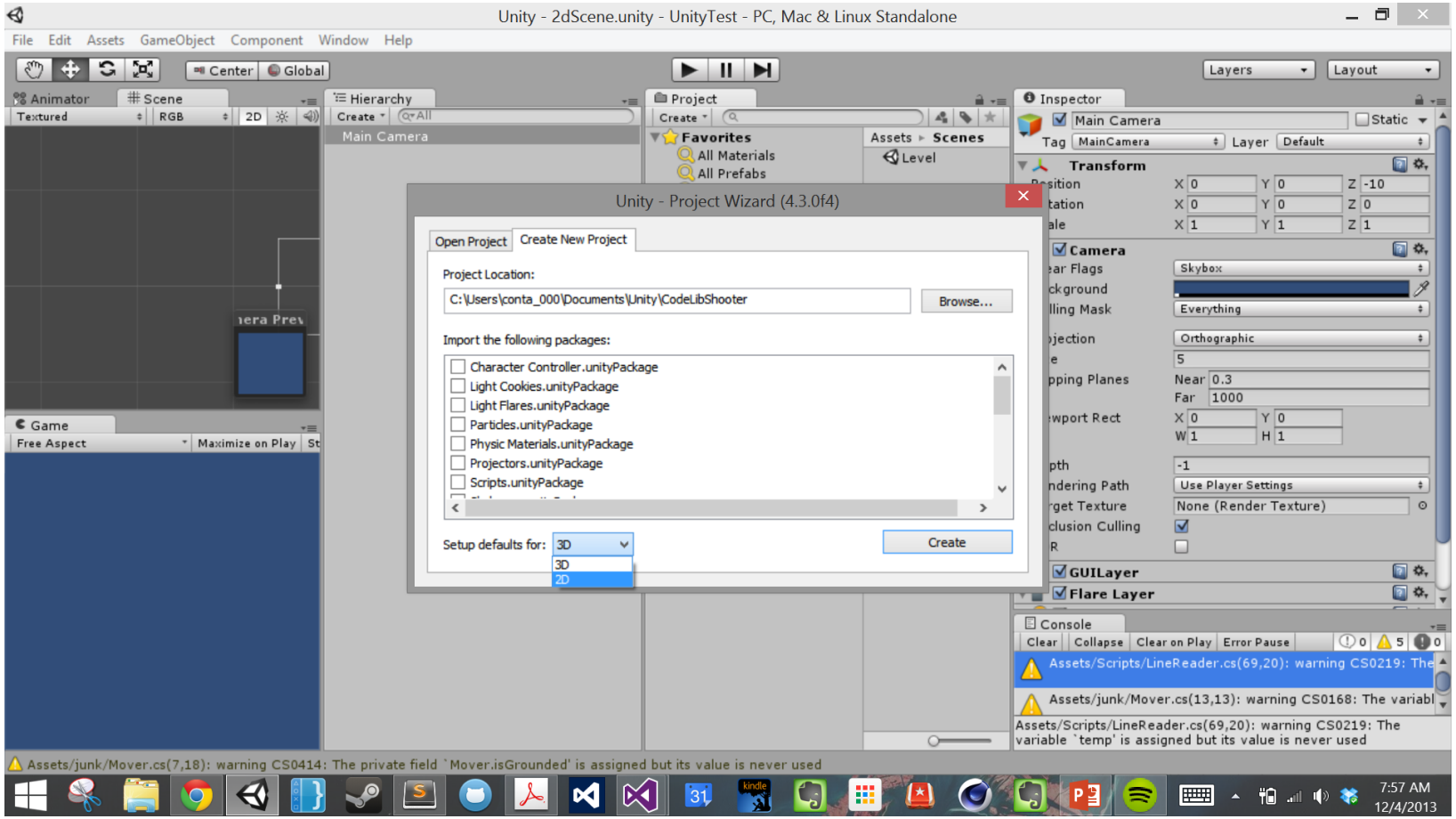


# World space vs Object space

Objects have their own axis and that is different than the world axis, which is always fixed in one direction and. Be warned forward for an object might not be forward for the world.



# Making a 2D project



# Sprites

2D Game Objects – Make a ship and a bullet

- Hierarchy tab > Create > Sprite
- Can contain many frames
- Can animate
- Can respond to physics and lights
- Can interact with only set sprites

# GUI Text

Flat onscreen text you can change with a script

Has font, color, position, size, scale, rotation, and all other goodies

Can animate in

Can NOT interact with in game objects. Aka you can’t shoot at your GUI text

# Prefab

- Saved game objects. You save these for creation later. You can create these objects (Instantiate) them dynamically with code.

- GameObject b = Instantiate(bullet, position, rotation) as Game Object;

- Nice because you can make it and forget it - it's done. To make a prefab drag an object from the Hierarchy to your Project tab. It will now turn blue in the hierarchy tab and get a cube icon next to it in the Project tab.

# Question time!

- What is a Game Object?
- What is a Component?
- Is Unity left or right handed?
- What is a prefab?
- Bonus points! Key command for rotate?

# The final piece of the puzzle. Scripts

- Scripts make components do things over time. Scripts are themselves components
- Scripts have access to anything you can see in the inspector
- Scripts let you LINK game objects together. It’s honestly the magic sauce.

# Unity has 4 main functions

```
void Awake() {
        //if the game object is in the hierarchy, even if the component isn't enabled and visible, this will run.
        // use for initialization and sending messages.
}
void Start () {
        // I  run when an object is put on screen and enabled
}
void Update()
{
        // every frame (variable rate)
}
void Fixed Update ()
{
        //for physics and at a fixed rate
}
```

# For loops

```
for(int i = 0; i<20; i++)
{
        //the code in this will 19 times.
}
```

Just a simple loop to do things. It says make an int and raise the value every time you run through the loop until it reaches an exit condition. You can think of it like a book mark in a book. You just keep moving it until you reach the end of the book.

Example game: clickingGame