12. Create a new Scene “terrain” and save it
13. GameObject > 3D Object > Terrain

Unity’s Terrain editor
• islands
• topographical landscapes
• Mountains
• And more
Textures

Textures should be in the following format to enable ‘tiling’

Square and the power of two

128 x 128
256 x 256
512 x 512
1024 x 1024

Shaders control the rendering characteristics of textured surface
Prefabs

pre-fabricated objects

Prefabs store a game object together with its components (transforms, appearances, scripts, etc.) and configurations for easy duplication/reuse.

- trees
- bullets
- characters, and anything else

Unity makes it easy to move around a world interactively (either in a first person or third person perspective) using prefabs.
Prefabs

Object-oriented instances can be **Instantiated** at run time

At **run time** a script can cause a new object instance to be created (instantiated) at a given location with a given set of properties

Prefabs allow functional game objects to be reused in scenes or imported into other projects as external assets.

The First Person Controller
First Person Controller

20. Assets > Import Package > Character Controller

Project Window > Standard Assets folder
FP Character > Prefabs > FPController
drag the FP Controller onto your scene
delete the main camera
Preview the game

explore the terrain / look around with your mouse
move with WASD or the arrow keys / jump with the space bar
First Person Controller

20. Assets > Import Package > Characters

Project Window > Standard Assets folder

FPS Character > Prefabs > FPSController

drag the FPS Controller onto your scene

delete the main camera

Preview the game

explore the terrain / look around with your mouse

move with WASD or the arrow keys / jump with the space bar
Scripting

MONO compiler
Scripts can be written in
  - JavaScript
    Majority of introductory tutorials are written in JavaScript

C#
Unity can be integrated with the Microsoft Visual Studio editor, to get full benefits of code completion, source version control, intergration, serious developers work in C#

BOO (like Python)
Smaller development in this
Scripting

scripting is Unity's most powerful tool
gives you the ability to customize objects
control how they behave in the environment
• how to create and attach JavaScript scripts to objects in Unity
• Intro to the development environment MonoDevelop

Variables
Functions
Triggers
Collisions
Sounds
Colors
JavaScript vs C#

**JavaScript**

```javascript
#pragma strict

var myInt : int = 5;
function Start ()
{
    myInt = MultiplyByTwo(myInt);
    Debug.Log (myInt);
}
```

**C#**

```csharp
using UnityEngine;
using System.Collections;

public class VariablesAndFunctions : MonoBehaviour
{
    int myInt = 5;
    void Start ()
    {
        int myInt = 5;
        Debug.Log (myInt);
    }
}```
You can use both C# and JavaScript in one project! (one way communication only)

<table>
<thead>
<tr>
<th>My Scripts Folder (Outside)</th>
<th>Standard Assets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiled last</td>
<td>Compiled first</td>
</tr>
<tr>
<td>Script</td>
<td>Script</td>
</tr>
<tr>
<td>Script</td>
<td>Script</td>
</tr>
<tr>
<td>script</td>
<td>Script</td>
</tr>
<tr>
<td>JavaScript</td>
<td>C#</td>
</tr>
</tbody>
</table>
JavaScript Variables

- A variable is a storage location and an associated symbolic name (an identifier) which contains some known or unknown quantity or information, a value

- variables are used to store information about any aspects of a project’s state
JavaScript Variables

begin with a lowercase letter
no special characters, numbers, (#, %, etc.)
cannot contain reserved keywords such as “if”, “while”, etc.
case sensitive
descriptive
no spaces
Declaration/ Type/ Initialization

var myVarBool : boolean = true;
var myVarInt : int = 10;
Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float</td>
<td>0.75</td>
</tr>
<tr>
<td>Int</td>
<td>10</td>
</tr>
<tr>
<td>String</td>
<td>“Hello”</td>
</tr>
<tr>
<td>Boolean</td>
<td>true / false</td>
</tr>
</tbody>
</table>

```javascript
var myVarBool : boolean = true;
var myVarInt : int = 10;
Var myFloat : float = 1.4;
```
Creating scripts in Unity

- Project menu > Create > JavaScript
- Main Menu > Assets > Create Javascript
- Project window > RMC > Create > JavaScript
- Inspector > Add script
- Name the script in the Project/Assets window

- Assign the script to an object (drag and drop)
- Run and test
- Fix compiler errors
Creating scripts in Unity

- Project menu > Create > JavaScript
- Main Menu > Assets > Create Javascript
- Project window > RMC > Create > JavaScript
- Inspector > Add script
Creating scripts in Unity:

- Project menu > Create > JavaScript
- Main Menu > Assets > Create Javascript
- Project window > RMC > Create > JavaScript
- Inspector > Add script
Creating scripts in Unity:

• Project menu > Create > JavaScript
• Main Menu > Assets > Create Javascript
• Project window > RMC > Create > JavaScript
• Inspector > Add script
Creating scripts in Unity

• Project menu > Create > JavaScript
• Main Menu > Assets > Create Javascript
• Project window > RMC > Create > JavaScript
• Inspector > Add script
Creating scripts in Unity

Creating scripts in Unity

- Project menu > Create > JavaScript
- Main Menu > Assets > Create Javascript
- Project window > RMC > Create > JavaScript
- Inspector > Add script

```javascript
#pragma strict

var myInt : int = 5;

function Start ()
{
    myInt = MultiplyByThree(myInt);
    Debug.Log (myInt);
}

function MultiplyByThree (number : int) : int
{
    var ret : int;
    ret = number * 3;
    return ret;
}
```
All compiler errors have to be fixed before you can enter playmode!
Creating scripts in Unity

- Project menu > Create > JavaScript
- Main Menu > Assets > Create > Javascript
- Project window > RMC > Create > JavaScript
- Inspector > Add script

All compiler errors have to be fixed before you can enter playmode!
Functions

Function is a collection of statements to perform a task.

Methods

Functions are blocks of code which are written once and can then be reused as often as needed. begin with an uppercase letter.

```
function FuncName ()
{
    statement1;
    statement 2;
}
```
JavaScript Functions

Calling a function:

FuncName();

myInt = MultiplyByThree(myInt);
Function Parameters

```csharp
function MultiplyByThree (number : int) : int {
    var ret : int;
    ret = number * 3;
    return ret;
}

Calling a function – myInt = MultiplyByThree(myInt);
```
Functions

Default functions

Start ()
executed only once before gameplay begins
helpful for initialization

Update()
executed every frame
for as long as the gameplay continues
Functions

Collections of tasks

Methods

Functions are blocks of code which are written once and can then be reused as often as needed. begin with an uppercase letter

```
function Start ()
{
    myInt = MultiplyByThree(myInt);
    Debug.Log (myInt);
}
```
Functions

var myInt : int = 5;

function Start ()
{
    myInt = MultiplyByThree(myInt);
    Debug.Log (myInt);
}

function MultiplyByThree (number : int) : int
{
    var ret : int;
    ret = number * 3;
    return ret;
}
Arithmetic Operators

+  addition
-  subtraction
/  division
*  multiplication
++ increment
-- decrement
%  modulus
Functions

1) Create 3D object cube
2) create new Javascript "rotateCube"
3) Assign the script to the cube (drag and drop)

```javascript
#pragma strict
var speed = 5.0;

function Start () {
}

function Update () {
    transform.Rotate(0, speed*Time.deltaTime, 0);
}
```
Functions

4) Change the value of var speed in the Inspector window (35)
5) Play and test
Triggers and Collisions

Triggers are methods to detect collisions
Triggers are useful for triggering other events in your project
  teletransportations
  automatic door openings
  displaying messages
  changing levels
  responsive events
  and many more
4) select the game object in the Hierarchy window
   click on the little gear on the top right corner of the script property
   select “remove component”
5) Create new script “triggerScript”

```javascript
var target : collider;
function OnTriggerEnter(cubeTrigger : collider)
{
    if (cubeTrigger == target)
    {
        print("Collision");
    }
}
```
Triggers and Collisions

6) Assign script to our cube
7) Check property “Is Trigger” in the Inspector
8) Create 3D plane
9) Import Character Controller Package
10) Drag FPC controller to the scene
11) Drag and drop the FPC from the Hierarchy window onto the variable Target in the Inspector
Triggers and Collisions

checks if the position of the FPC
intersects with the position of the trigger zone (the cube)
prints out “Collision”
Triggers and Collisions

To add a counter to collision
Checks how many times collision happened

```
var target : Collider;
private var counter : int = 0;

function OnTriggerEnter(cubeTrigger : Collider)
{
if (cubeTrigger == target)
{
  counter = counter + 1;
  print("Collided: " + counter + " times!");
}
```
Triggers and Collisions

To add a counter to counter checks how many times collision happened

```csharp
var target: Collider;
private var counter: int = 0;

function OnTriggerEnter(cubeTrigger: Collider) {
    if (cubeTrigger == target) {
        counter = counter + 1;
        print("Collided: "+counter+" times!");
    }
}
```
Triggers and Collisions

to create an invisible trigger zone

Select the object >
Inspector > remove Mesh Renderer Component

The object will be invisible but still allow collision detection
Sounds

Supported Audio Formats

MPEG layer 3 .mp3
Ogg Vorbis .ogg
Microsoft Wave .wav
Audio Interchange File Format .aiff / .aif
Ultimate Soundtracker module .mod
Impulse Tracker module .it
Scream Tracker module .s3m
FastTracker 2 module .xm
13) Import new Asset (sound effect/s)
14) Add Audio Source to the Cube (Inspector>Add Component >Audio Source)
15) Uncheck button “Play On Awake”
16) Drag sound effect to the Inspector > Trigger Script >My sound
Sounds

var target : Collider;
private var counter : int = 0;
var mySound : AudioClip;

function OnTriggerEnter(cubeTrigger : Collider)
{
if (cubeTrigger == target)
{
  audio.PlayOneShot(mySound);
  counter = counter + 1;
  print("Collided: " + counter + " times!");
}
}
13) Import new Asset (sound effect/s)
14) Add Audio Source to the Cube (Inspector>Add Component>Audio Source)
15) Uncheck button "Play On Awake"
16) Drag sound effect to the Inspector > Trigger Script > My sound
Sounds

13) Import new Asset (sound effect/s)
14) Add Audio Source to the Cube (Inspector>Add Component>Audio Source)
15) Uncheck the button “Play On Awake”
16) Drag sound effect to the Inspector > Trigger Script > My sound
13) Import new Asset (sound effect/s)
14) Add Audio Source to the Cube (Inspector > Add Component > Audio Source)
15) Uncheck the button "Play On Awake"
16) Drag sound effect to the Inspector > Trigger Script > My Sound
13) Import new Asset (sound effect/s)
14) Add Audio Source to the Cube (Inspector > Add Component > Audio Source)
15) Uncheck the button "Play On Awake"
16) Drag sound effect to the Inspector > Trigger Script > My Sound
17) Create new material and add transparent/diffuse shader
18) Add material to the cube
19) Modify the script to add new material:

```plaintext
private var orange : Color = Color(0.8, 0.4, 0.0, 0.3);

renderer.material.color = orange;
```
Color Systems

CMYK – The Subtractive System

RGB – The Additive System

CMYK Color Model

RGB Color Model
Color Cube
RGBA Color

Representation of RGBA colors
Values for red, green, blue and alpha are floating point values with a range from 0 to 1

Alpha component (a) defines transparency
alpha of 1 is completely opaque, alpha of zero is completely transparent

Black  RGBA is (0, 0, 0, 1)
  blue  RGBA is (0, 0, 1, 1)
Gray   RGBA is (0.5, 0.5, 0.5, 1)
Clear  Completely transparent. RGBA is (0, 0, 0, 0)
RGBA Color

- Black RGBA is (0, 0, 0, 1)
- Blue RGBA is (0, 0, 1, 1)
- Gray RGBA is (0.5, 0.5, 0.5, 1)
- Clear Completely transparent. RGBA is (0, 0, 0, 0)

Magenta?
Yellow?
cyans?
17) Create new material and add transparent/diffuse shader

18) Add material to the cube

19) Modify the script to add new material:

```csharp
private var orange: Color = Color(0.8, 0.4, 0.0, 0.3);

renderer.material.color = orange;
```
Colors

20) Create new material and add transparent/diffuse shader
21) Create new 3D object - cylinder
22) Add material to the cylinder
23) Modify the script to add new material:

```javascript
var cylinderMaterial : Material;

cylinderMaterial.color = orange;
```

24) Assign cylinderMaterial to the var cylinderMaterial in the inspector
20) Create new material and add transparent/diffuse shader
21) Create new 3D object - cylinder
22) Add material to the cylinder
23) Modify the script to add new material:

```plaintext
var cylinderMaterial : Material;
tcylinderMaterial.color = orange;
```