Unity 2D Basics Cheat Sheet

by Become A Game Developer (become) via cheatography.com/67591/cs/17006/

Editor's Interface

Scene view	Build the game world, interact with game objects
Game view	Preview and play game (pressing Play in Toolbar)
Inspector	Show and modify game objects' components' properties
Hierarchy	Game objects in game
Project	Assets available
Toolbar	Bar with buttons at the top. Contains transform tools, play controls, layers and layout
Assets	Files (scripts, textures, models, prefabs)
Console	Contains debug logs and errors
Tags, Layers, Sorting Layers	Open from Edit > Project Settings > Tags and Layers. Tags are identifiers for Game Objects. Game Objects can belong to Layers. Objects in last Layers are rendered above the others. SpriteRender Components can belong to Sorting Layers, which define the rendering order for sprites. Camera Components can see or not Sorting Layers by setting the Culling Mask

Game Objects and Components

Game Object	Basic entity in Unity. Can be a 3D or 2D object, a particle or audio or video source, a UI element, or an empty object. Game Objects are just containers for Components. Scripts can be attached to Game Objects, to define their behavior and properties. Game Objects in your scene are represented in the Hierarchy
Component	Basic entities that implement functionalities inside Game Objects
Component in the Inspector	Each Component has a small header bar with: Turn down arrow, Icon, (De)activate checkbox, Reference book (opens online manual), Preset button, Options gear (allows to copy and paste Components). Under the bar are all the Component's properties

Game Objects and Components (cont)

Prefab	Blueprint for Game Objects. You can make a Prefab out of a Game Object. The Prefab will be like a "-model" from which you can instantiate new identical copies of that object in your game. Modifying the Prefab properties will modify all Game Objects instantiated from it
Parent /Child	Any Game Object can have other Game Objects as children. The Transform of a child Game Object will be relative to the parent's Transform. If you make a Prefab out of a Game Object with children, all the hierarchy will be copied. You can see parent/child relationships in the Hierarchy
Usage	
Create new Game Object	Right click on the Hierarchy > select the Game Object type

By **Become A Game Developer** (become)
cheatography.com/become/

becomeagamedeveloper.github.io/site

Published 13th September, 2018.
Last updated 13th September, 2018.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Game Objects and Components (cont)

Game Objects' name	Set the name from the Inspector (upper part), or from slow double click on the object in the Hierarchy
Tag	Assign custom Tags to Game Objects from the Inspector (upper part)
Add a child	In the Hierarchy, drag a Game Object over another
Add Component	Inspector > Add Component
Create Prefab	Drag the Game Object from the Hierarchy to the Project window
Create Game Object from Prefab	Drag the Prefab from the Project to the Scene view or the Hierarchy

Game Objects and Components (cont)

Modify a Prefab	If you select a Prefab from the Project, and you modify its properties/components, all objects of that type will be modified. On the contrary, if you modify a single Game Object, you can then, from the Inspector (upper part) click on Prefab: Apply button to modify the Prefab
Deactivate Game Object	Click the tickbox in the upper part of the Inspector

Game Objects and Components (cont)

Reference Game Object in the Inspector	If you define public `GameObject` or Component (ex: `Transform`) variables in a script, they will be visible as properties in the Inspector (under the Script Component). You can assign Game Objects to these variables by dragging a Game Object from the Hierarchy to the field in the Inspector. If the variable is of type `GameObject`, you reference the whole Game Object. If it is of some Component type, instead, you will reference that Game Object's Component directly instead

By **Become A Game Developer** (become)
cheatography.com/become/

becomeagamedeveloper.github.io/site

Published 13th September, 2018.
Last updated 13th September, 2018.

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Game Objects and Components (cont)

Reference Prefab in the Inspector	The same way you reference a Game Object or a Component in a Script variable from the Inspector, you can drag a Prefab from the Project window to reference it. This is useful for instantiating copies of the Prefab later on
Instantiate Prefab	To instantiate a Prefab from a Script: define a variable of type `GameObject` or of some Component type. Reference the Prefab from the Inspector. You can now use the `Instantiate()` function (see API section)

Basic Game Objects and Components

Basic Game Objects

Sprite	2D graphic Game Object. Contains SpriteRenderer component, that manages the rendering of the texture. If you add 2D Colliders and/or 2D RigidBody Components, the Sprite will behave like a physical object

Basic Game Objects and Components (cont)

Camera	Contains a Camera component and an Audio Listener. Gives the window through which you can experience your game's world. In a new scene, there is always a Main Camera already present. You can parent a Camera to a Game Object to follow it (or setup a script that continually sets the Camera's position to the Game Object's position)

Basic Components

Transform	Determines position, rotation and scale. It is always present
Sprite-Renderer	Display an image (Sprite property). You can create and set a Sorting Layer to define which sprite is rendered above and which below when two sprites overlap
Camera	Capture and display the world. Has several options, such as background default color, field of view. In Culling Mask you can set what layers to render and what to ignore. Game Objects belonging to ignored layers won't be seen
Script	Defines custom properties and behavior of a Game Object

2D Physics

Physics Components

Rigidbody 2D	Places an object under control of the physics engine, giving it a Body Type (see below), a mass, a new position (overriding the Transform's one), a velocity, an angular velocity, a Material (defining drag and bounce), and allowing it to be affected by forces (gravity, drag, impulse). Attaching a Rigidbody to a Sprite makes it behave in a physically convincing way
Collider 2D	Defines the shape for the purpose of collisions. Can be edited by clicking "Edit Collider". Can be set to "Trigger" to emit events

Body types

Dynamic	Body designed to move. Collides with any body type. Can be affected by forces
Static	Doesn't move (infinite mass). Collides with Dynamic bodies. Gives back forces when colliding
Kinematic	Designed to move (only via function calls). It moves accordingly to its velocity, but it's not affected by forces. Collides only with Dynamic bodies

Mechanics

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

2D Physics (cont)

Spatial coordinates	The position of a body is identified by a point (Vector3 with 3 coordinates in 3D space, Vector2 in 2D space). In games, the X axis grows from left to right, while the Y grows from top to bottom (it's reversed)
S = V * T	A body with velocity V moves by S in a timestep T
V = A * T	A body with an acceleration A increases its velocity by V in a timestep T
F = m * A	Applying a force F to a body with mass m causes an acceleration A on it
Static drag	If there is a static drag D on a surface, a body cannot move unless you apply a force F > D to it
Dynamic drag	If there is dynamic drag D on a surface, a body moving on it will constantly have a force D *opposed* to where it's moving
Gravity	If there is a gravitational acceleration g, a body will have a downwards acceleration of g

2D Physics (cont)

Trajectory	Curve on which a body moves. A projectile (Angry Bird) with just an initial velocity and in a gravitational field will "draw" a parabula shape. The projectile lands farther if the initial velocity vector was at 45°
Angular mechanics	When dealing with rotations, simply substitute: position with angle, velocity with angular velocity, acceleration with angular acceleration, force with torque, drag with angular drag. The laws stay the same

Scripting

What is a script	A script is a file containing code (usually C#) that defines the properties and the behavior of a Game Object
Adding a script	Add a Script to a Game Object from Inspector > Add Component > New script
Editing a script	Double click on the script in the Inspector. It will be open with your default external editor (Visual Studio, Monodevelop). Then change the script and save it

Scripting (cont)

Script contents	A script usually has import statements in the upper part, and then the code of a class (with the same name of the file), that contains variables and methods
Set variable from Inspector	From the Inspector you can set values for public variables defined in Scripts. For numbers and strings, type directly in. For `GameObject` or any Component types, drag Game Objects or Prefabs from the editor to the field
Compiling	Once a script is saved, Unity automatically (re)compiles it. It may take some time (wait for the loading gif in the bar below to disappear)
Debugging	Compile-time errors and debug logs (outputs of Debug.Log()) are shown in the Console

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Unity 2D Basics Cheat Sheet

by Become A Game Developer (become) via cheatography.com/67591/cs/17006/

Scripting (cont)

Accessing classes from other scripts	If a Script defines a `public` class, you can use that class as a reference in any other Script. Ex: the script Enemy contains a reference to the class Player because it needs to chase it
Documentation	From the text editor, select a term and press Ctrl + '

C#

Syntax

statement ;	End every statement with a semicolon
`using` *namespace*	Include namespace, making new classes available
`class` *name* : *father* { }	Define class (inheriting from *father* class). A class is a blueprint that you can use to instantiate an object: a special variable that contains its own variables (members) and functions (methods)
`public` *field*	Make a member or method visible in the Inspector and accessible from other scripts
`private` *field*	Deny access from other scripts
`// comment`	One-line comment
`/* multi-line comment */`	Multiple line comment

Types

`bool`	`true` or `false`
`int`	Integer number
`float`	Decimal number. Floats always end in *f*. Ex: `4.5f`
`string`	Text
someType`[]`	Array containing objects of type *someType*

Variables

int `a;`	Declaring a variable
a `= 5;`	Assigning a value
ClassName b;	Declaring a reference (variable that can contain an object)
Component Class myComponent;	For every Component, there exist a class with the same name that you can use to refer to it (ex: `Transform`)
b `= new` *ClassName*`();`	Instantiating an object

C# (cont)

```
int[] myList = new int[5];
```

```
myLlist[0] = 9;
```

Methods

```
retType Name(type1 arg1, ...) {
  body
  return X; }
```

```
object.me thod( )
```

Control flow

```
if(condition){
  code
} else if (condition) {
  code
} else {
  code}
```

```
while(condition){
  code}
```

object.va riable	Accessing an object's member variable
`null`	Value for null reference (a variable referring to no object)

```
for(int i=X; i++; i<N){
code}
```

```
foreach(type x in myList){
code}
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

C# (cont)

Operators

+ - * / % ? ! ++ --	Operators (4 operations, modulus, ternary conditional, not, increase, decrease)
< > == != <= >=	Relational operators (lesser, greater, equal, different, less or equal, greater or equal)
cond1 && cond2	'and' operator. True only if both conditions are true
cond1 || cond2	'or' operator. False only if both conditions are false

In *italics* generic or sample terms

Code Flow and Events

Code Flow	Scripts do not run in the traditional manner, looping until they complete a task. Instead, Unity runs the main Game Loop (think of it as a `while` loop where continuously the following things happen: external input is taken, the game state is updated, objects may be created and destroyed, physics and graphics computations are run, and a new frame is rendered on the screen). When events of a certain type happen, Unity passes control to Scripts by calling the corresponding function. These are called Event Functions

Code Flow and Events (cont)

Event Functions	Callback functions that are called by Unity when certain events occur. Event Functions are provided as methods of the MonoBehaviour class, from which the classes in every Script inherit
Trigger Collider	Checkbox you can tick in a Collider Component. If active, the object will emit a *trigger event* when in contact with something. Triggers are used for non-physical collisions (e.g. detecting when someone enters)

MonoBehaviour Event Functions

`Awake()`	Called once before everything else
`Start()`	Called once after all Awakes, before any Update

Code Flow and Events (cont)

`Update()`	Code that changes the position, state, behavior of objects in game. It is called before each frame is rendered. Updates happen at every iteration of the Game Loop, therefore each update may take a different time. The variable `Time.d elt - aTime` always contains the duration of the last update iteration
`FixedU pdate( )`	Called before each physics update. The physics engine updates in time steps of fixed duration, therefore you don't have to correct for deltaTimes when moving things inside FixedUpdate. Place physics calls inside this
`OnMous e***()`	Called when there is a mouse event. *** can be: `Down`, `Enter`, `Exit`, `Over`, `Up`, `UpAsButton`

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Unity 2D Basics Cheat Sheet

by Become A Game Developer (become) via cheatography.com/67591/cs/17006/

Code Flow and Events (cont)

`OnColl isi on** *2 D(C oll ision collision)`

`OnTrig ger *** 2D( Col lider other)`

Unity game loop: https://docs.unity3d.com/uploads/Main/monobehaviour_flowchart.svg

Input

Input Manager	Access in the editor by Edit > Project Settings > Input. Contains the properties of the Axes

Input (cont)

Axes

Called when the object is involved in a collision. *** can be: `Enter`, `Stay`, `Exit` (contact is made, held, or broken). The parameter contains info about the collision

Called when the object is involved in a collision, only if the object's collider is configured as a Trigger. *** can be: `Enter`, `Stay`, `Exit`. The parameter is the other object's collider

API

MonoBehaviour

`MonoBe haviour`

`gameObject`

`tag`

`transform`

`GetCom pon ent<Type>()`

`GetCom pon ent s<Type>()`

`Destro y(O bject obj, float t = 0.0f);`

virtual directions ("Horizontal", "Vertical", "Jump", "Fire1",...) that represent possible inputs. Each Axis has a name, and one or two buttons that are mapped onto the *Positive direction* and the *Negative direction* (e.g. "Fire1" only has a button for the Positive input, since there is no concept of *firing backwards*)

Getting Input	When a player presses an axis button, Unity will set the axis state to a value between -1 and 1 (-1 negative, 1 positive, 0 when there is no input). Get the input by querying Axes. Alternatively, you can query using button names (Keys)
`value = Input.GetAxis("Horizontal");`	Retrieves the current state for the *"Horizontal"* Axis
`value = Input.GetKey("a");`	Returns `true` if the user is holding down the key *"a"*

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Cheatography

Unity 2D Basics Cheat Sheet

by Become A Game Developer (become) via cheatography.com/67591/cs/17006/

API (cont)

`Instan tia te( Object original)`	Clones a GameObject or Component and returns the clone. If you clone a Component, the whole GameObject it is attached to will also be cloned

GameObject

`GameObject`	Base class for all entities in Unity Scenes
`tag`	Tag of this Game Object
`transform`	Transform of this Game Object
`SetAct ive (bool value)`	Activate/-deactivate this Game Object
`GameOb jec t.F ind Wit hTag(" Tag ")`	Static method that finds and returns the first Game Object with tag *Tag*

Component

API (cont)

`rotation`	
`Rotate (float xAngle, float yAngle, float zAngle)`	
`Transl ate (Ve ctor3 transl ation)`	

Vector

`new Vector2(x,y)`	
`new Vector3(x,y,z)`	
`v + u, v - u`	
`v * 5, v / 5`	
`target.position - player.position`	
`v.magnitude`	
`v.norma lized`	
`v.x`	

Debugging

`Debug.L og (*"m ess age ");`	

API (cont)

Rigidbody2D	Rotation as a `Quaternion`
`mass`	Rotate around X, Y, Z axis
	Moves position. Ex: `trans rwa rd* Tim e.d elt`
`AddFor ce( Vector2 force)`	Create new 2D vector
	Create new 3D vector. Tha even in 2D! (But you shou
	Sum/subtract two vectors
	Multiply/divide a vector by a
	Vector representing the dis
`AddTor que (float torque)`	Transforms
	Vector's length
`MovePo sit ion (Ve ctor2 posit ion)`	Vector with the same direc
	Access X component (sa
	Prints a messaget to the C

Full scripting API: https://docs.unity3d.com/ScriptReference/index.html
Manual: https://docs.unity3d.com/Manual/index.html
2D guides: https://unity3d.com/learn/tutorials/s/2d-game-creation

`Component`	Base class for everything attached to GameObjects. For every specifc component there is a class (with the same name as the Component), inheriting from this. Ex: `Transform`

Transform

`position`	Position as a `Vector3`

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com