Week 7: Objects and C# Classes (Part 3)

Welcome to Week 7!

Presentation Slides

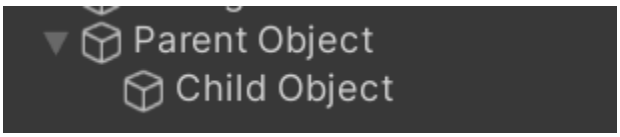
Worksheet (document)

Worksheet (PDF)

Table of Contents

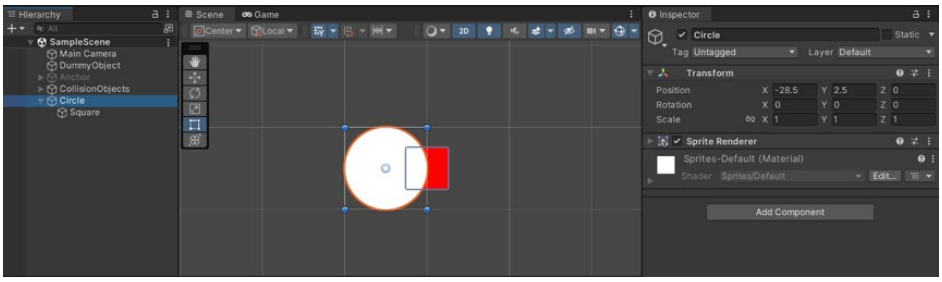
7.1 – Unity Physics

7.1.1 – Object Hierarchy

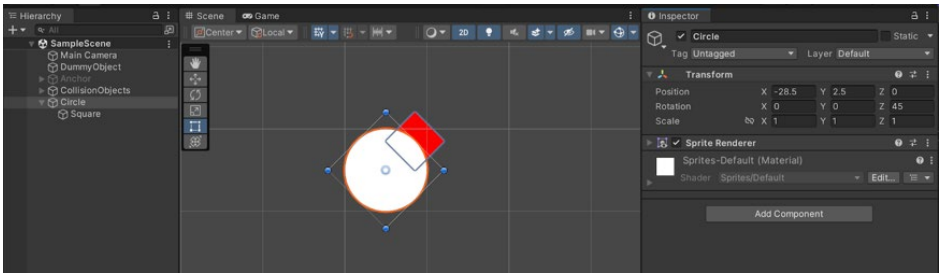


In Unity, objects in a scene can be added to the hierarchy of another object. To do this, click-and-drag the object onto the other object.

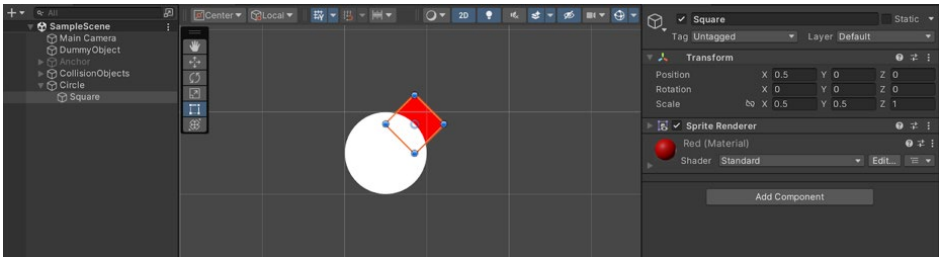
The object that was moved is called a *child* of the other object, which is called the *parent* object. To see why this is useful, consider the following Circle-Square hierarchy.



If the rotation of the Circle object is changed to 45 degrees, the position of the Square object is also changed.



The Square object is a *child* of the Circle object, so any changes to location, rotation and scale on the Circle object will also indirectly affect the Square object without changing its components.



Hierarchy vs. Inheritance

The **class** inheritance mentioned in Week 6 might sound related to object hierarchies because of its name, but the two are not related in any way.

7.1.2 – Transform

Unity handles the locations, rotations and scales of objects in a scene through the **Transform** component. **Transform** contains several properties for handling these values.

- **Transform parent** – The **Transform** component on the parent object for the **GameObject** that the current **Transform** component is associated with
- **Vector3 position** – The absolute location of the **GameObject** in the scene
- **Vector3 localPosition** – The location of the **GameObject** relative to the **Transform** component on its parent object
- **Vector3 eulerAngles** – The absolute rotation of the **GameObject** in the scene

- **Vector3 localEulerAngles** – The rotation of the **GameObject** relative to the **Transform** component on its parent object
- **Vector3 localScale** – The size of the **GameObject** relative to the **Transform** component on its parent object

If the **GameObject** has no parent object, then the relative variants of the properties are the same as the absolute variants.

Transform in the Inspector

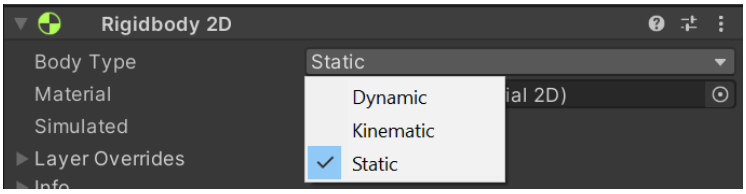
The *Position*, *Rotation* and *Scale* values in the **Transform** component when viewing it in the *Inspector* window of the Unity editor correspond to the **localPosition** , **localEulerAngles** and **localScale** properties.

7.2 – Rigidbody2D

The **Transform** component's intended use was for 3D scenes, and trying to use it to move and rotate objects in a 2D scene can be unintuitive and confusing.

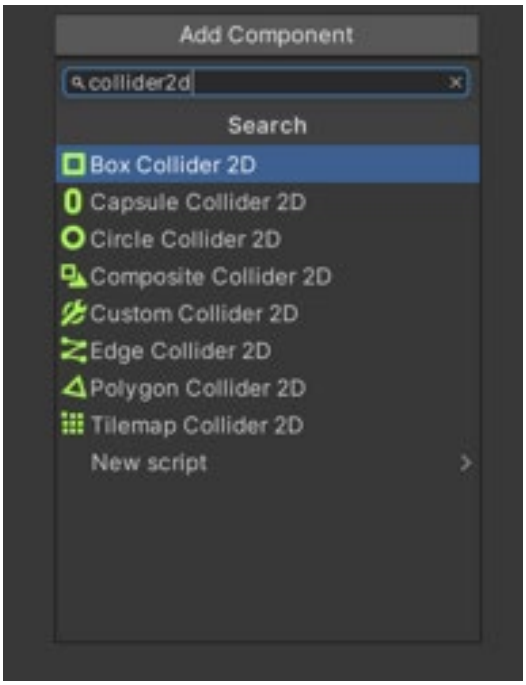
For this course, the **Rigidbody2D** component will be used to address this concern while also enabling more "realistic" physics in Unity. **Rigidbody2D** contains properties which act as abstractions over the **Transform** properties, plus other properties unique to it.

- **Vector2 position** – The absolute location of the **GameObject** in 2D space
- **float rotation** – The orientation of the **GameObject** in degrees
- **Vector2 linearVelocity** – The direction and speed of the **GameObject** in units per second
- **float angularVelocity** – How fast the **GameObject** is rotating in degrees per second
- **float gravityScale** – How strongly gravity affects the object from **0.0f** to **1.0f**



All objects using **Rigidbody2D** components are affected by gravity by default. To prevent this, as is the case for things like level geometry, change the *Body Type* from *Dynamic* to *Static*.

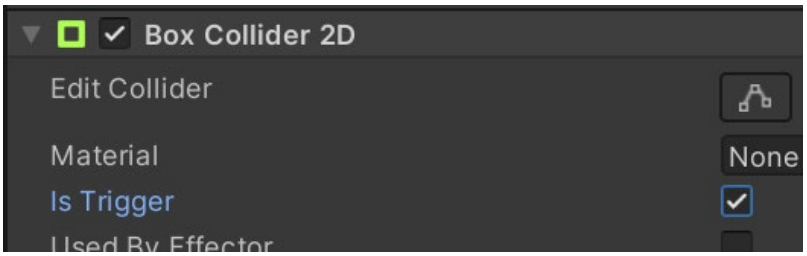
You are encouraged to try using **Rigidbody2D** on objects in your project's scene. In doing so, you may notice that the objects pass through each other. To implement collision detection between objects, another component is required.



7.3 – Collision Detection

7.3.1 – Collider2D

The **Collider2D** component allows a **Rigidbody2D** object to collide with other **Rigidbody2D** objects. Trying to search for **Collider2D** in the Unity editor won't show the component itself[1] and will instead show the components to the right. However, each of these components inherit from **Collider2D** , so they are still relevant.



Objects with a **Collider2D** component are called *colliders* or *triggers* depending on the value of the **isTrigger** property.

---

[1] **Collider2D is an abstract class , which is why it doesn't appear in the editor. The abstract data type modifier is outside the scope of this course, so it won't be included for brevity.**

7.3.2 – Colliders

Colliders are **Collider2D** components whose **isTrigger** property is **false** . Any **Rigidbody2D** that would need to be stopped by floors, walls and ceilings[2] should use a *collider* component.

Furthermore, for **Rigidbody2D** objects whose *Body Type* is set to *Static*, collisions with other objects will not affect the position of the object.

To listen for when collisions occur between objects, use the methods provided by **MonoBehaviour** .

- **void OnCollisionEnter2D(Collision2D collision)** – Executes on the first physics update when another object has collided with the object for this **MonoBehaviour** .
- **void OnCollisionStay2D(Collision2D collision)** – Executes every physics update while another object is colliding with the object for this **MonoBehaviour** .
- **void OnCollisionExit2D(Collision2D collision)** – Executes on the first physics update when another object is no longer colliding with the object for this **MonoBehaviour** .

To get the other object, use the **collision.gameObject** variable. Keep in mind that these methods get executed on both objects at the same time.

---

[2] Floors, walls and ceilings themselves also use *collider* components.

7.3.3 – Triggers

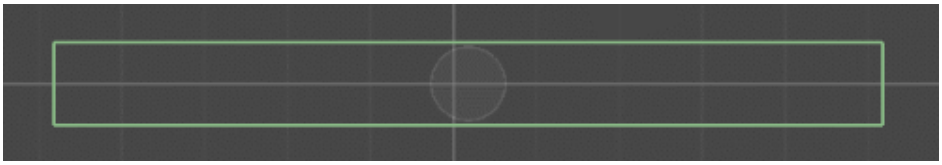
Triggers are **Collider2D** components whose **isTrigger** property is **true** . Triggers do not affect the position and velocity of objects and are typically used for things like collectable pickup detection, death planes in platformer games as well as playing cutscenes when the player has reached a specific area in a level.

To listen for when an object has collided with a trigger, use the methods provided by **MonoBehaviour** .

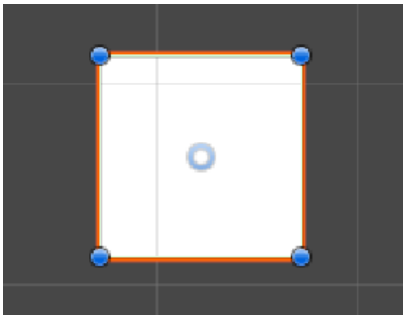
- **void OnTriggerEnter2D(Collision2D collision)** – Executes on the first physics update when another object has collided with the trigger on the object for this **MonoBehaviour** .
- **void OnTriggerStay2D(Collision2D collision)** – Executes every physics update while another object is colliding with the trigger on the object for this **MonoBehaviour** .
- **void OnTriggerExit2D(Collision2D collision)** – Executes on the first physics update when another object is no longer colliding with the trigger on the object for this **MonoBehaviour** .

Much like the methods for colliders, the **collision.gameObject** variable can be used to get the other object.

For both colliders and triggers, you can see their bounds in the *Scene* window as green edges (when not exactly on the edges of a shape)



or red edges (when they are).



7.4 – Example

In the slides for Week 7, the code for an example of a trigger object and collider objects was shown. This section will cover this example in more detail.

```
public class TeleportCollider : MonoBehaviour
{
    private void OnTriggerEnter2D(Collision2D collision)
    {
        collision.gameObject.SendMessage("OnReset");
    }
}
```

The **TeleportCollider** script is very simple; it just uses **OnTriggerEnter2D** to send a "message" to the other object.

GameObject.SendMessage(string method) is used to indirectly execute methods on all components on a **GameObject** . A method with a name equal to **method** must exist on at least one component.

Sending Messages for Methods That Do Not Exist

A separate definition of **SendMessage** exists that can specify that the method does not have to exist on the components of a **GameObject** . In the case of the script in the example, that would be changed to the following:

```
collision.gameObject.SendMessage("OnReset", SendMessageOptions.DontRequireReceiver);
```

```
public class TeleportBackToOrigin : MonoBehaviour
{
    private Rigidbody2D rb;
    private Vector2 originPosition;
    private bool teleport;

    private void Start( )
    {
        rb = GetComponent<Rigidbody2D>( );
        originPosition = rb.position;
    }

    private void FixedUpdate( )
    {
        if (teleport)
        {
            teleport = false;

            // Update the physics variables
            rb.position = originPosition;
            rb.linearVelocity = Vector2.zero;
            rb.angularVelocity = 0.0f;
            rb.rotation = 0.0f;
        }
    }

    public void OnReset( )
    {
        // Unity message
        teleport = true;
    }
}
```

The **TeleportBackToOrigin** script may seem complex at first glance, but it is also simple.

The **OnReset** method gets executed by the **SendMessage** method call from the other script, which sets a boolean field to **true** .

The object then gets teleported within **FixedUpdate** , a method provided by **MonoBehaviour** which gets executed at a consistent rate for physics calculations. The teleportation sets the position of the **Rigidbody2D** of the object to the position that was stored into a field by the **Start** method, then sets its velocity, angular velocity and rotation to zero.

Vector2.zero is a property which gets a 2D vector whose X- and Y-components are zero.

7.5 – Slides

Note: The colors for text and certain elements in the following images of the slides have been modified to produce lighter colors when printing with the *Black and white* Color setting and do not reflect the colors used in the actual slides.


Objects and
C# Classes
UA LITTLE ROCK EAC
57


60
Unity Physics
62
Rigidbody2D
64
Collider2D
58


GameObject Hierarchy
An object in the scene can have other objects added to it
Click-and-drag the object onto the other object
Parent Object
Child Object
59

Unity Physics

The Transform component contains **properties** for the location, orientation and size of a GameObject

- Transform parent – The Transform component of the parent object
- Vector3 position – Absolute location in the **scene**
- Vector3 localPosition – Relative location to the Transform of the parent object
- Vector3 eulerAngles – Orientation of the object in degrees
- Vector3 localEulerAngles – Orientation of the object relative to the Transform of the parent object

Unity Physics

The Transform component contains **properties** for the location, orientation and size of a GameObject

- Vector3 localScale – Size of the object relative to the Transform of the parent object

Rigidbody2D

An abstraction over physics-related variables on an object in a **scene**. Relevant **properties** are:

- Vector2 position – The location in 2D space
- float rotation – The orientation in degrees
- Vector2 linearVelocity – The direction and speed of the object in units per second
- float angularVelocity – How fast the object is rotating in degrees per second
- float gravityScale – How strongly gravity affects the object

Rigidbody2D Physics...?

- Objects with Rigidbody2D components cannot interact with each other by default

63

Collider2D

An abstraction over collision events between objects in a **scene**.

- "Colliders" – Used to stop movement of an object
- "Triggers" – Doesn't affect movement

64

Collision Detection

No extra steps are required to make colliders work

65

Collision Detection

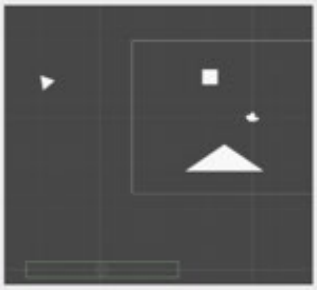
MonoBehaviour has methods that are executed when two objects with Collider2D components collide with each other

- void OnCollisionEnter2D(Collision2D collision) – collision.gameObject has collided with the object for this MonoBehaviour
- void OnCollisionStay2D(Collision2D collision) – collision.gameObject is still colliding with the object for this MonoBehaviour
- void OnCollisionExit2D(Collision2D collision) – collision.gameObject is no longer colliding with the object for this MonoBehaviour

Questions?

Triggers

Non-solid colliders typically used for "area detection" events



Triggers

MonoBehaviour also has methods that are executed when an object with a Collider2D component collides with a trigger Collider2D

- void OnTriggerEnter2D(Collision2D collision) – collision.gameObject has collided with the trigger on the object for this MonoBehaviour
- void OnTriggerStay2D(Collision2D collision) – collision.gameObject is still colliding with the trigger on the object for this MonoBehaviour
- void OnTriggerExit2D(Collision2D collision) – collision.gameObject is no longer colliding with the trigger on the object for this MonoBehaviour

Triggers

Trigger object

```
public class TeleportCollider : MonoBehaviour
{
    private void OnTriggerEnter2D(Collision2D collision)
    {
        collision.gameObject.SendMessage("OnReset");
    }
}
```

Square object

```
public class TeleportBackToOrigin : MonoBehaviour
{
    private Rigidbody2D rb;
    private Vector2 originPosition;
    private bool teleport;

    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        originPosition = rb.position;
    }

    private void FixedUpdate()
    {
        if (teleport)
        {
            teleport = false;

            // update the players variables
            rb.position = originPosition;
            rb.linearVelocity = Vector2.zero;
            rb.angularVelocity = 0.0f;
            rb.rotation = 0.0f;
        }
    }

    public void OnReset()
    {
        // debug message
        teleport = true;
    }
}
```


Triggers

- GameObject.SendMessage( ) executes the parameterless **method** with the same name as its string parameter on all **components** on the GameObject, if it exists

```
public class TeleportCollider : MonoBehaviour
{
    private void OnTriggerEnter2D(Collision2D collision)
    {
        collision.gameObject.SendMessage("OnReset");
    }
}
```

7.6 – Exercises

Reinforcement

The reinforcement section is used to check the understanding of the information that was presented in the PowerPoint/document. Please answer the questions in your own words.

R — 7.1 What are parent and child objects?

R — 7.2 Why are object hierarchies used?

R — 7.3 What component is used for the position, orientation and size of an object in a scene? What are its properties, and which of them appear in the *Inspector* window of the Unity editor?

R — 7.4 What component is used for 2D physics in Unity? What are its properties?

R — 7.5 What component is used for collision detection between objects? What are its properties?

R — 7.6 What is the difference between a collider and a trigger?

R — 7.7 What methods does **MonoBehaviour** provide for colliders?

R — 7.8 What methods does **MonoBehaviour** provide for triggers?

Project

The project section's tasks are used in work toward creating the final Flying Bird game project for the end of the course.

P — 7.1 Create a script that gives a **Rigidbody2D** upward velocity when the spacebar is pressed.

P — 7.2 Create a script that sends a message to the Unity console when a **Collider2D** on its object collides with something.

P — 7.3 Expand the script from 7.2 to also indirectly execute a method on the other object's components.

P — 7.4 Create a script that deletes an object when it collides with a trigger.