Collision detection in video games: How best to interact with other game components?

COMP130 - Software Engineering

1706966

March 20, 2018

This paper looks into some of the different ways a collision detection system designed for use in a video game, can interact with different components of the code and best practice for setting up these interactions. Some insight into how this can affect the end user will be given. Then on the paper will take a look at how some of the industry standard development kits handle collision detection system cohesion. Afterwards looking at design patterns that can be used to give a base when implementing a system that isn't a pre built one. Going on to summarise how to use this knowledge to get the best communication between collision system and other components.

1 Introduction

Collision detection systems themselves have played a part in video games since the start. Being used to simulate the world in which they are creating.

Having a computer know that one object has come into contact with another, then use the information it has gathered from that contact to inform another component. Forms the basis of a collision detection system. For example in a game of pong, as the ball hits a paddle, this is detected and its information is gathered and sent to another component that changes how the ball now moves based on the information given. This is a very rudimentary example, most games now have much more complex systems, making it more important to streamline how systems interact. Having only the data that is needed sent to the other component saves on memory and time. If a system is broken it can cause dissatisfaction for the end user. Things might occur such as a character falling through the floor, or being stuck on something that they can't see because the detection isn't fine enough. There are reasons a bad collision system could cause dissatisfaction that might not be as obvious that it's being cause by the collision system. If a system is poorly optimised it can slow down the game causing lag issues and poor frames per second. Some development kits come with ways of handling collision detection built into them, such as unity and unreal, we will be taking a look at how these work. However when writing a programme in basic language you have a few different options available. Some of which being design patterns that can be taken as a base and worked into your code to fit your needs. There are also some open source collision detection libraries that can be downloaded and used.

2 Industry Development Kits

2.1 Unity

Within Unity you can give objects either colliders or rigidbodies (The colliders and rigidbodies have to be set on the object and given measurements.) both of these give the object the ability to call "OnCollisionEnter" when they collide with other objects that have one of the aforementioned properties. A collision class is passed when OnCollisionEnter is used, the class contains information about contact points impact velocity and more. If you are not planning to use this information you can leave out a parameter to save unnecessary calculation[1] A downside to this method is that this has to find any code that is calling for it on collision. Alternatively you are able to use "OnTriggerEnter" which does a similar thing but is called when a collider enters a set trigger. Only the fact the event has been triggered is sent, and that information is sent only to the collider that hit the trigger and the trigger itself[2] Meaning a lot less data is passed in comparison to OnCollisionEnter, and its being sent to very specific locations. Unity has supplied us with some very easy to implement ways of checking for collisions, but it does have its caveats. Both objects need to be either colliders or rigidbodies. One of them must be a rigidbody in the case of the trigger event. Data that these events supply is also restricted, having basically an all or nothing approach.

2.2 Unreal

In Unreal Engine the set-up of how the colliders interact with each other is similar to unity, all object you want to be able to collide need to be given a static mesh first. A static mesh can be wrapped around an object a finely as required. A more precise wrap will require more resources. Once an object has been given a static mesh and been set to collision, you are then able to designate the type of collision. There are many different types along with custom grouping systems, but a few main ones will be listed. First is blocking, both objects block the other from passing, but no event is generated from the collision. This is used for when you want to stop an object but don't need to know anything about it. Next we have Hit Events, similar to blocking neither object can pass through the other but this time we have an event activate. An event can be triggered on either object or both, and the event will tell any code attached to it that the object has hit another object. Information about the other object can be passed through. Lastly there is Overlap Events, these activate when the two objects overlap one another, so they don't block like before but instead pass through. The objects can either ignore this or send data similar to the Hit Event data[3]. Unreal shows more easy customisation when it comes to setting up collision on objects and how they send data to other area's. However if you want to inform some other than one of the objects colliding you would need to code that inside one of the objects.

3 Design Patterns

4 Libraries

.

5 Conclusion

As seems to always be the case with coding, the best way to do something is always very dependant on your needs. It's turned out to be frustratingly impossible to say which is best for all occasions. But following these best practices you can be sure you're on the right track. Only send information where you need it if possible, and limit that information to the minimum required for the component receiving the data. Having your detection system be more customisable lets you be more specific in what information is gathered on collision, in turn letting you dispatch that information more accurately.

References

[1] U. Technologies, "Unity - scripting api: Collider.oncollisionenter(collision)," 2018. [Online]. Available: https://docs.unity3d.com/ScriptReference/Collider.OnCollisionEnter.html

[2] ——, "Unity - scripting api: Collider.ontriggerenter(collider)," 2018. [Online]. Available: https://docs.unity3d.com/ScriptReference/Collider.OnTriggerEnter.html

[3] E. Games, "Collision overview," 2018. [Online]. Available: https://docs.unrealengine.com/en-US/Engine/Physics/Collision/Overview