2023

Deterministic Replay

COMPLEX GAME SYSTEMS

CONNOR MILLS

Table of Contents

Overview

The modular system that I will be creating is a deterministic replay system. Deterministic replay is a technique that provides the user with deterministic executions of computer programs while nondeterministic factors are still present.

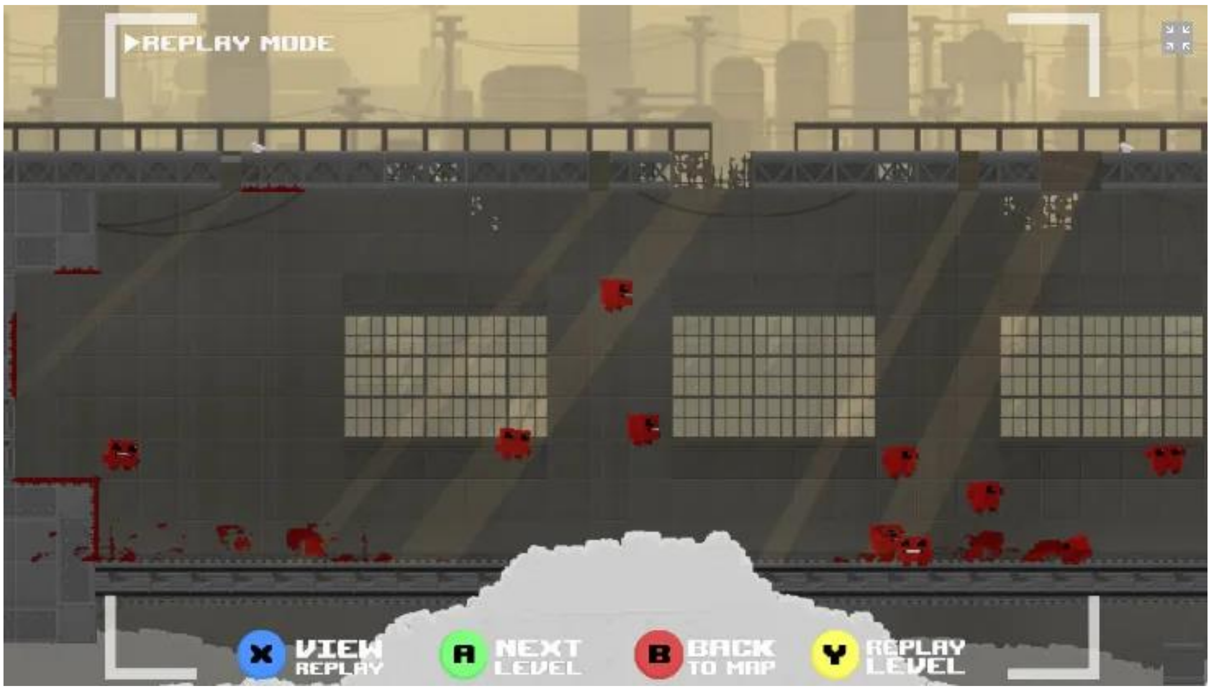
The system that I will create will record the position, rotation, animation states, and other relevant information of the player-controlled character and any other moving object. That information can then at a later date be used to replay the player's movement and other objects, displaying a ghost of them that follows the exact same path as they took when they were recorded.

The Objective

I am designing it to allow development teams access to an easy to use and accessible tool that they can use to add an extra feature to their game. Deterministic replay systems are a great way to enhance a game and to also add a new competitive aspect to single player games. It can be used in multiple different applications for lots of different games. Some examples are:

In a single player racing game, each of the player's attempt at the course could be recorded and then the fastest time recorded could then be played when the player starts next. This allows the player to compete against an opponent pushing themselves to get better. Or the player could view the replay and see where they did well and where they did well, allowing them to perfect their gameplay.

Another application is in a platformer game where all attempts are recorded and then when the player beats the level all the attempts are then played together showing the player each time they died. This can be seen in the game Super Meat Boy made by Team Meat where at the end of a level each attempt made to beat it is then simultaneously shown in a replay.



Third Party Libraries

The system will be created as a Unity package and will use Unity's libraries. I will be using MemoryStream and the read and write functions from System.IO to store the game data used in the replays.

Mathematical Operations and Advance Algorithms

Operations

The system is very light in mathematical operations. The only mathematical operations that occur is when the scripts that are recording calculate the difference between the current frame and the previous frame and save it to the queue. Also, when the scripts that are reading from the queue get the data that was saved and add it to the ghost object.

When the system is recording, the scripts that handles the collection of the object's data will hold onto the previous frame's data and each frame it will checks it against the current frame's data. The script calculates the difference between the two frames by subtracting the previous frame's data from the current frame's data to get the change between the two frames and then store it onto the queue.

Current Frame - Previous Frame = Difference

Once the system has finished recording and is now playing, the scripts that controls the ghost objects will pull from the queue in order and will get the difference that is stored in the queue and then apply it the ghost object.

Algorithms

An unoptimized replay system will take up a lot of storage as it will record every piece of data every frame no matter what. This means that even if nothing changes between frame it will still record the data. To fix this issue I have decided to use a MemoryStream in my system. This will allow me better control over what information I can store and in what order.

To optimize my system, I have decided to check every frame if the data has changed from the previous frame and if so, record the data else the data will not be recorded. The method I will be using is to record one single bit either as 0 or 1 for if the data has changed and then save the relevant data afterwards. This will mean that if a piece of data did not change then the bit will be 0 and no further data will be saved, cutting out unneeded data and saving on storage.

On the reading side of the system, the program will step through the MemoryStream using a BinaryReader. It will first look at the bit and check if it is 0 or 1, and if it is 1 it will then read the relevant data from the stream. But if it is 0 then it will move on and start the process again by checking the bit for the next section of data.

Modular Design

The system I will create will be designed to be modular. It will be a Unity package that any user can download and import into their project. The system will contain presets the user can use, and they have the option of creating their own custom recording and reading scripts.

I will create presets that can be used in most games that need a ghost replay feature. These presets will include the recording and reading scripts that are needed to create the ghosts. The presets will account for what type of information needs to be recorded. Such as position, rotation, scale, animation states, and other relevant data.

To make sure users can create their own custom recording and reading scripts I will provide several things. A step-by-step guide for how to create your own custom scripts will be provided in the documentation. Fully fleshed out and well documented derivative scripts that can be used. As well as several example scripts.

Integration

As stated earlier the system will be exported as a Unity package. This package can be easily downloaded by any user to then import into their projects. The user can then just select on of the provided presets or create their own. Drag the recording script onto whichever game object they want to record, add a slimed down version of the game object to the reading scripts, create the replay manager and link up the events that their want to use and they are all set.

References

C# - Queue (no date) *TutorialsTeacher.com*. Tutorials Teacher. Available at: https://www.tutorialsteacher.com/csharp/csharp-queue (Accessed: May 1, 2023).

How to create a Replay System like in Super Meat Boy using Unity (2022) YouTube video. added by Trever Mock [Online]. Available at: https://www.youtube.com/watch?v=ilOQstDnX2I&ab_channel=TreverMock [Accessed: May 1, 2023].

Engel, T. (2020) *Creating a replay system in Unity*, *Kodeco*. Edited by A. Rames, M. Moser, and B. MacKinnon. Kodeco. Available at: https://www.kodeco.com/7728186-creating-a-replay-system-in-unity [Accessed: May 1, 2023].

System.IO Namespace (no date) *Microsoft Learn*. Microsoft. Available at: https://learn.microsoft.com/en-us/dotnet/api/system.io?redirectedfrom=MSDN&view=net-7.0 [Accessed: May 1, 2023].

Morgan, M. (2011) *Super meat boy is your ticket to Classic 8-bit gaming*, *Wired*. Conde Nast. Available at: https://www.wired.com/2011/01/super-meat-boy-is-your-ticket-to-classic-8-bit-gaming/ [Accessed: May 2, 2023].

Super Meat Boy. (2010). Xbox 360, Windows, OS X, Linux, PlayStation 4, PlayStation Vita, Wii U, Nintendo Switch [Game]. Asheville, North Carolina, United States: Team Meat.

Wagner, C. (2004) *Developing your own replay system*, *Game Developer*. Informa PLC. Available at: https://www.gamedeveloper.com/programming/developing-your-own-replay-system#close-modal (Accessed: May 3, 2023).

Trackmania. (2020). Windows [Game]. Saint-Mandé, Paris, France: Ubisoft.