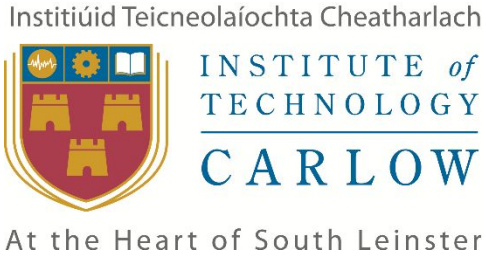


Computer Games Development
SRS and Project Report
Year IV

Greg Cahill
C00216328
03/05/2020

INSTITUTE of TECHNOLOGY CARLOW
Institiúid Teicneolaíochta Cheatharlach

Faculty of Science

Open-Book and Remote Assessment Cover Page

Student Name: Greg Cahill

Student Number: C00216328

Lecturer Name: Dr Lei Shi

Module: Project II (fyp)

Stage/Year: Fourth Year

Date: 03/05/2020

Declaration

This examination/assessment will be submitted using Turnitin as the online submission tool. By submitting my examination/assessment to Turnitin, I am declaring that this examination/assessment is my own work. I understand that I may be required to orally defend any of my answers, to the lecturer, at a given time after the examination/assessment has been completed, as outlined in the student regulations.

[Declaration form to be attached]

Contents

Acknowledgements

I would like to thank the following people who assisted in completing this project including;

Dr. Lei Shi as my project supervisor for helpful advice and counseling in regards to what to focus on, and where to find relevant information in regards to this project

Project Abstract

Effective techniques to mitigate a poor quality internet connection, in the context of gaming at least, is an important issue as the sending of packets has to be relatively fast. Due to this, the finer details may be lost in transit as a result of packet loss.

This is usually fine, but in sub par connections, it causes all players frustration and a poor impression of a product. My task I set out to do with this project was to compare and contrast two common methods of dealing with latency in terms of networking in games.

The two I decided to focus on for this game are Rollback and Delay based

I want to see how implementing these two into the same ‘game’ can change how the game feels to play. I also want to test the relative ease of developing one or the other so that in a company sense, which is more beneficial to use.

My research has shown that delay based is far more widely used, but rollback implemented well can lead to much better results for all players. I want to see if this project can show both why Delay based is more commonly used, and if the effectiveness of rollback is worth the ‘cost’ of implementing it.

Project Introduction and Research Question

The question I wish to tackle is how to mitigate the issue of players with poor connections attempting to play a game with a large online community that they can't participate in due to said poor connection

Net-code as a whole is pretty good, with fighting games being 1v1 running a peer to peer model (lock step) as connecting to a server in this case would be too slow for the updates required/ More updates equals more precision in regards to how often data is sent between the players,

One of the biggest issues that I would like to focus my priority on is if in a high latency environment, introduce options for players with poor connections to still enjoy the game.

Rollback netcode from my research seems to solve this exact issue, but the vast majority of developers ignore it, in favour of delay based , or other methods that may fit the game they are developing more. By comparing and contrasting the Rollback and Delay based netcodes, I ultimately want to get to the bottom of why one is more common than the other, despite the apparent advantages to Rollback

Background

My current research is from what is currently implemented in the world today, a great example of rollback net code is what google stadia's 'negative latency' essentially is. The stadia, in its need to get around the absurd amount of data to be streamed, predicts the players inputs given context of a situation (e.g. A player is falling, the last time they fell they jumped out by pressing this button, they are probably pressing this button now) so it does that action ahead of time to cut out that slight pause that would throw the player out of their immersion.

Of course, it's not a flawless system, as when a wrong action is assumed, the handling of that is the crux of the issue. This is where the difficult part lies , at least from my research of listening to developer guests on podcasts, and reading excerpts from the developers themselves talking about how they were trying to implement their own version of rollback into their games. (For example, Guilty Gear: Strive developed by Arc system works are changing from delay based to their own rollback system, and while fans are pleased, The devs seem...less than keen on implementing it.)

My background research on Delay based is a lot more general, as it's a dime a dozen, in terms of implementation. Due to how common it was, finding specific information wasn't quite what I was looking for so I looked into how players felt about it. Players from built up areas with good internet didn't notice anything, whereas players with poor internet, or in areas such as russia that often get placed in european servers said it left a lot to be desired, and in some cases stopped them from playing the game at all, which is bad for both a healthy playerbase, and sales via word of mouth , or poor reviews.

Literature Review

"An analysis of continuous consistency models in real time peer-to-peer fighting games"
-see references below

This study tackled what I'm tackling very similarly, although there method was to compare different known techniques in regards to the online match itself. They came to the conclusion that rollback netcode is the superior choice as latency increases. This falls in line with my research as well, My compare-and-contrast will hopefully fall in line with the conclusions reached here, and reinforce each other.

"GGPO ROLLBACK NETCODE"

While not a true study, this is something that was made available for public use very recently, and has been used in the industry for years prior and has been praised for being very good. It is the rollback netcode that is what was described in the above reference. This is what appears to be the gold standard for rollback netcode, as developers and high level players now saw there's no excuse for a game to have poor netcode with this out. This is a fantastic framework to build around.

Project Description

My aim for the study is to simply compare and contrast all the possible advantages and disadvantages of these two methods, both in terms of how effective the methods deal with poor quality connections, and in terms of implementing them into a project.

My procedure in how to tackle the issue is to document the implementation process and compare the final outcomes of the two methodologies. The easiest way to test the outcomes are purely

from a visual and gameplay feel. If the game feels ‘bad’ to play, or you are actively fighting the game itself to simply play it , then that is a very easy way to see that downside.

My technical learning in terms of this project was how I overestimated my skills, and my ability to tackle completely new ideas. In terms of implementing a peer to peer system , and rollback in particular. I learned that I really should focus on what I know I can do, as opposed to what I find interesting. In terms of technical achievement, there wasn't a whole lot pushing boundaries, but i'm happy with the attempt I made to try to simulate an online game in an offline environment through the use of threading. Simple, but got the point across.

Personal learning , is I have developed an understanding for the sheer amount of work that goes into developing the online component of a game, and the decisions that have to be made along the way to either save time, or money and manpower in companies minds. Personal achievements would fall under the fact that despite what I was up against, I gave it a shot anyways. Once I knew what was involved, it was quite daunting on how to tackle it.

But, I buckled up and tried (and failed in some cases) to show my focused down questions upsides and downsides in my comparison

Overview

-Philosophy

Point 1-
This project I’m working on will not shake up anything in terms of the gaming industry, this comparison has been tackled by people smarter than me, more talented than me and in greater detail. I’m now keenly aware of my limits, and will work within them.

Point 2-
Use SDL for the game we are testing, this is simply one that is easiest to port, and we have tested int the past developing online components for (albeit, with a server as opposed to peer to peer)

Final Point-
My design , with 100% certainty , must be mine. I cannot, and will not take from peoples existing ideas. I will try to understand their implementation, then develop my own. Plagiarism in any capacity is not welcome.

Common Questions

What is the project you are working on?

Effectively, what I'm working on is a small scale comparison using methodologies in use in the gaming industry at the moment commonly used to mitigate poor quality connections.
This project won't have a major game component attached to it, as ultimately whatever the game is shouldnt matter, whether implemented in the next call of duty , or a multiplayer snakes and ladders game, the results would be the same, or at least should be.

Why research this specific topic?

I grew up in the country in the middle of nowhere in county carlow, which is already the middle of nowhere in terms of Ireland. I experienced dial up connections until I was around 12 to 13, and only very recently got internet that can exceed download speed of 1 mb/s. Poor connections is a topic close to my heart, and one I genuinely wished to tackle.

What is the game part attached to this project?

The game portion is super simple, its a square that moves around the screen that the player controls via keyboard input. The crux lies in how the player moves in terms of updates, and how it feels to move them. Despite its simplicity , it does effectively get the point across of game feel matters, and with low quality connections, you will feel that hit.

What do I want out of this project

What I ultimately want out of this project is to test my skills as a programmer by drifting into *very* unfamiliar waters, and to come to a conclusion, both as a finished product and during the implementation process, why delay based seems to be the most commonly used method.

Define the Application

This application , at least what its supposed to be was a comparison of delay based and rollback based netcode. Due to my own lacking skills, the online component could not get off the ground, and an offline simulation was developed instead, but still roughly got the point across, especially the upsides and downsides during the development process.

What the product is currently Is a game that lets you move a square around on a screen using the keyboard. The delay based implementation is the more accurate of the two in terms of the

finished outcomes, but the rollback is not fully implemented due to my lacking skill set and technical know-how. Knowing what I now know, it may be easier to focus on certain aspects of it, but that is not what's in the finished product currently.

What the application is supposed to do

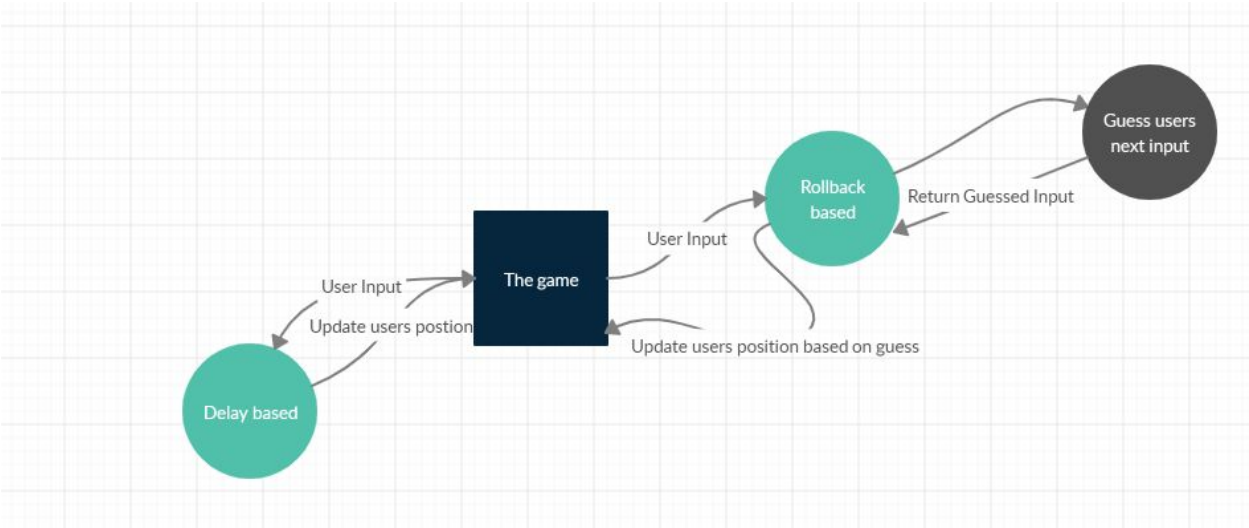
-Delay based approach
This demonstrates how a game can be smooth to play and control, should the milliseconds of lag/delay be an acceptable level, whereas as that number gets higher and higher, it suddenly becomes a challenge to move something as simple as a square around a screen, so in an actual fully fledged game, you can easily see how this would be a problem.

-Rollback based approach:
This was to guess a players appropriate input based off their last inputs, and should that guess be off when the actual input is received, its resets a couple frames back to the last correct state. The first part isn't too bad, but its the handling of the second part wherein the challenge lies, as should the guess be handled wrong even once, the entire game would be out of sync and then no one is having fun as it feels like decisions do not matter.

Built using SDL-
The convenience of SFML , while nice, it can leave a lot to be desired in terms of porting a project to other devices, or the handling of online components. While SDL is a lot more in depth and complex, the options you have with it far outweigh the convenience of SFML (Funnily enough, a similar comparison to delay based and rollback based.)

Who will use this application
The main user group for this application is developers asking the same question I asked, "which approach is the best to use for the game I am making", and hopefully with my research , they come to a much more informed decision that they won't regret down the road.

Context Diagram



Metrics

To gauge whether my project was successful, all I truly want is a comparison between the two methods , being delay and rollback based. The upsides and downsides to both will be accounted for, and while I have a preference in gameplay for rollback, I cannot blame a developer for choosing delay based for 95% of cases being adequate enough. This comparison should provide enough information for a developer to choose which method they wish to use , granting an informed decision.

Precedent for the application

Yes, and no. May reports and video series on youtube compare the two, but only by their finished products and how they feel from a players perspective. This comparison differs in the fact that as both a player and a developer, I can look into this using a different mindset completely that the comparison videos of exclusively players, or exclusively developers.

Design Manual

This application will be used by doing the following , simply using the arrow keys on the keyboard, and moving the square around the screen. You may comment out the rollback section, or the delay based section depending on which one you wish to use , the controls and game remain the same regardless of which one you choose.

There is no outright UI on screen, only the square itself, as the intention is to show off the two methods. UI could be added showing the current delay for delay based, and the arrow keys to move, but no much levels of feedback are needed, as the intention comes across in the game feel itself.

The delay based code and rollback based code will not interact with each other, as they aren't meant to be ran separately , bar that the only relationships worth mentioning in the project is that using multi threading, the potential out of sync nature that could come up is handled via the sleep methods, which as a by product also functions as an artificial , scalable delay.

Project Milestones

-implement the ‘easier’ of the two netcodes, delay based.

-Implement rollback, and a simple game of moving squares

-Test both methods using attempted peer to peer style connections

-Added later, create offline version of similar tests

-A simple game to demonstrate both in action.

Project Review and Conclusions

For starters, I feel like even though the finished product was vastly different from what I ultimately wanted out of the project, it still demonstrated the upsides and downsides to both delay based and rollback based coding styles.
However, far more went wrong than right. I overestimated my skill set, and underestimated the challenge in tackling a new topic that I had very little understanding of going into it. Rollback and peer to peer in particular went through so many iterations and changes, and in the final product, the online component was scraped entirely as I didn’t understand it well enough to develop my own way of handling it without directly plagiarizing someone else's work.
As stated above, the rollback would need to be finalised and cleaned up to a much higher degree, and peer to peer lockstep as a whole would have been nice to implement.

My advice for someone attempting a similar project in the future would be to know your limits. Even if you know what you don't know, understand your ability to also adapt and learn what you don't know. In a completely different sense, if I was telling the me of the past when I started this, I would have told him the choose a completely different topic, one we understood more deeply and could demonstrate our skills better.
I do believe the choices I made technology wise were correct, c++ and SDL are two things I have great experience in, so they were easy to use in the project as a whole which let me focus on the 'point' of the project itself.

References

Referenced Publication	Citation	Reference
Study	(**An analysis of continuous consistency models in real time peer-to-peer fighting games 2019**)	Authors - Martin Huynh and Fernando Valerino, An analysis of continuous consistency models in real time peer-to-peer fighting games 2019 http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1322881&dswid=-395
Website	GGPO rollback netcode	https://github.com/pond3r/ggpo
Website	Making a multiplayer shooter in c++	http://www.codersblock.org/blog/multiplayer-fps-part-1