FPSVR Design Manual

VR First Person Shooter Game + Game Console + VR headset

Group 20

E/16/173/394/396
CO227 Project

Content

1. Introduction

i. Overview

a. Virtual Reality Industry

b. Gaming industry

c. Opportunity

ii. Design

a. Solution

b. Overall Design

2. Game development

i. Platform

ii. Design

3. Network architecture

i. Overview

a. Network structure

ii. Security

a. User data security

b. Token verification

c. Anti Cheating mechanism

iii. Performance

a. Solution overview

b. Clock sync

c. Interpolation

d. Extrapolation

iv. Cloud (AWS) implementation

a. AWS choice

b. Implementation

4. Hardware design

i. Overview

a. Actions

b. Game controller (wired)

c. Game controller (Bluetooth)

d. VR headset

ii. Components

a. ADXL345 Sensor

b. ATTiny85V board

c. HC05 Bluetooth module

d. PushButton

e. 18650 Li-Ion battery

f. 18650 Battery Shield

iii. Sensor model circuit diagram

5. Testing

i. Sensor Calibration

ii. VR module Testing

iii. Smoothing movement

iv. Network testing

1. Introduction

I. Overview

a. Virtual reality industry

Virtual Reality (VR) is a computer-generated environment with scenes and objects that appear to be real, making the user feel they are immersed in their surroundings and it is perceived through a virtual reality headset. VR allows us to immerse ourselves in environments as if we were one of the characters.

Although this may seem futuristic, VR concept has been in use since the first computers and has been experimented in various use cases such as education, design, architecture, military and gaming.

The global virtual reality market size was valued at USD 15.81 billion in 2020 and is expected to grow at an impressive compound annual growth rate (CAGR) of 18.0% from 2021 to 2028 according to Virtual Reality Market Share & Trends Report, 2021-2028.

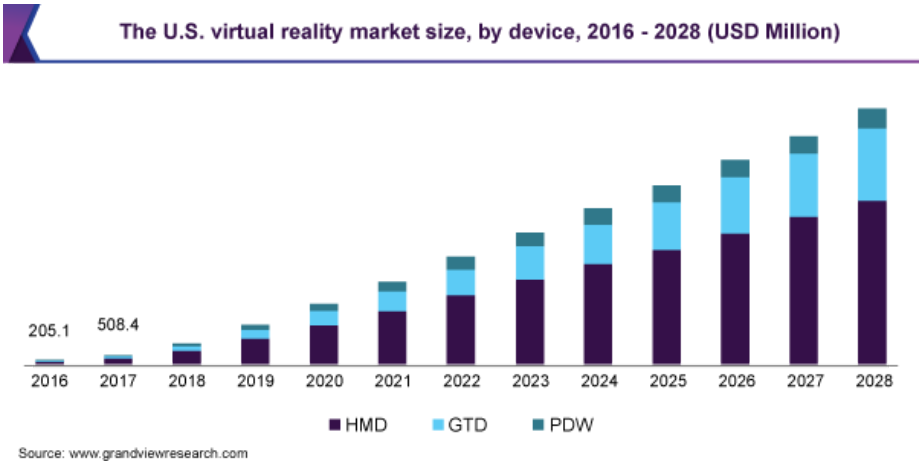


Figure 01 : US Virtual Reality Market Forecast

According to the same report, the consumer application segment accounts for nearly 20% market share. The consumer application segment is dominated by the gaming industry with over 80% dominance.

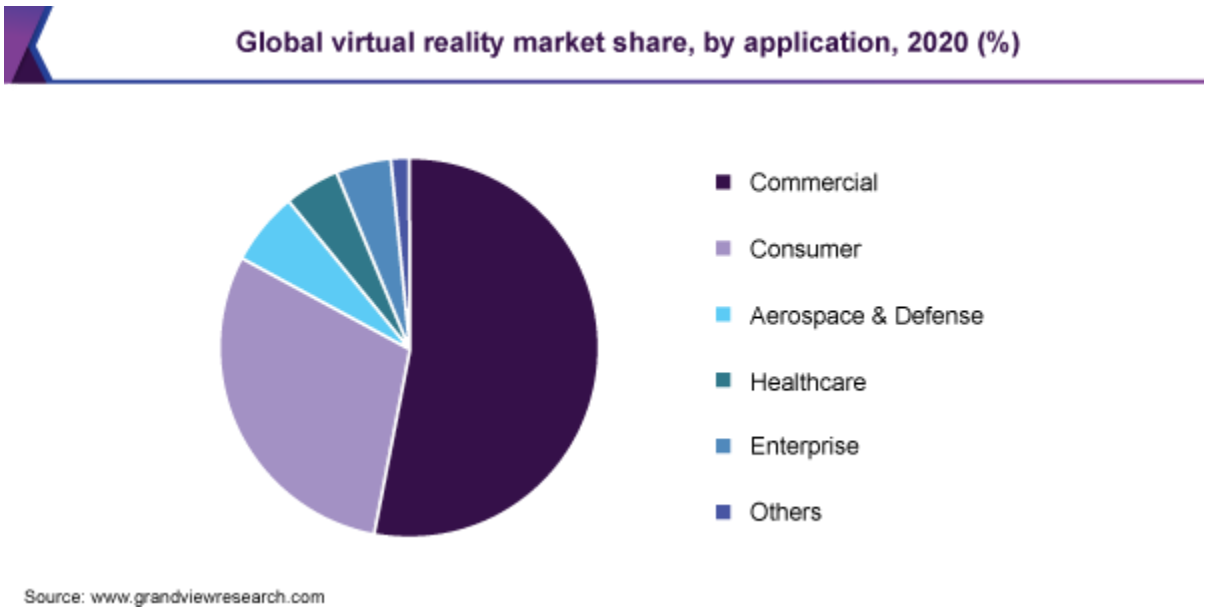


Figure 02 :Global Virtual Reality Marketby application

b. Gaming industry

We've all seen how much success the gaming industry has had in light of the COVID-19 pandemic. Due to their increased time at home, gamers have had the opportunity to download and purchase new games and consoles, and so much more.

According to IDC research 2020 on gaming, the gaming industry outperformed both movies and sports combined last year as the biggest moneymaker in entertainment. In fact, on a global scale, the gaming industry was valued at $162.32 billion in 2020. Based on the current growth rate, it's expected to reach a value of almost $300 billion over the next five years.

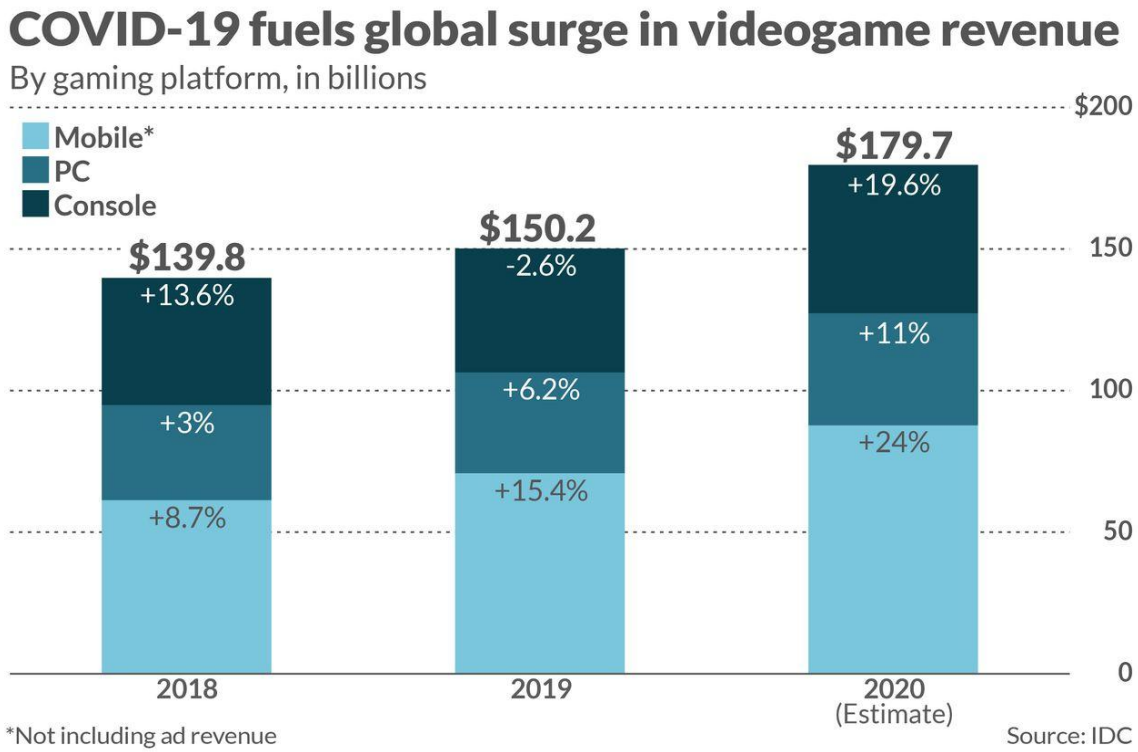


Figure 03 : Global surge in video game revenue

c. Opportunity

There are few dominant players in the VR industry such as Facebook/Oculus, HTC, Samsung VR, Microsoft holo etc.

However, all of those companies only target high end markets with built in screen high end headsets.

In the current market, the lowest price of a VR headset is roughly $299 which is clearly too high for the average consumer in developing countries like Sri Lanka.

There is a clear market gap for an entry level VR headset for consumers who are willing to experiment with VR gaming.

Further, VR based game development is currently booming and now is a good time to start on VR game development.

II. Design

a. Solution

FPSVR project is the starting step of the FPSVR VR gaming platform.

Currently, FPSVR package consists of the followings.

- VR multiplayer first person shooter game
- VR headset (smartphone screen compatible)
- VR game controller

The package will be appealing to the customer due to,

- Much lower price
- Free game updates.
- Future Free & paid game launches

b. Overall design

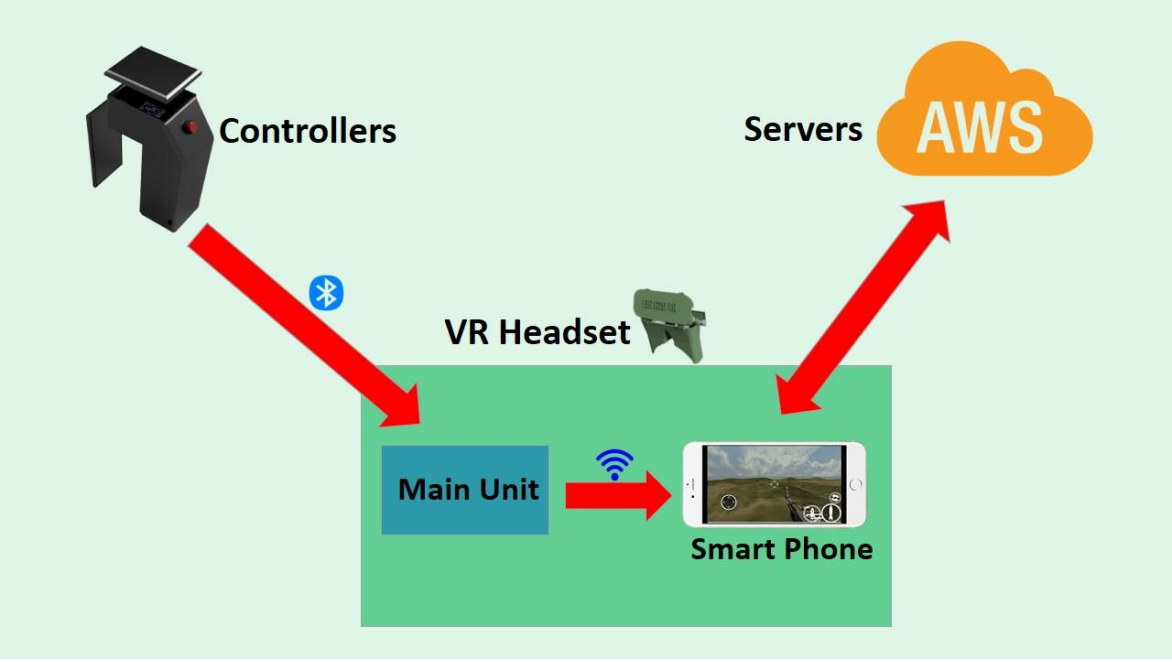


Figure 04 : Overall design

The sensor unit captures controller movements by the player and sent data to the playing device via bluetooth/wire. The game application process data and translates them into the game actions.

The smartphone connects to wifi and update player states in game servers on the cloud.

The servers handle authentication and data syncing to provide a multiplayer gaming experience.

2. Game Development

I. Platform

The game engine of choice for the FPSVR is the Godot game engine version 3.3.1.

Godot excels as an easy to use ,rapid game developing game engine with large community support.

The coding language gdscript is also identified as a simple, easy to use and very intuitive language with full support for development with the object-oriented API.

II. Design

FPVSR game is a multiplayer game designed using the dedicated server - clients architecture.The game can be run on Android phones that have support for VR (Android 4.1+) with gyroscope and accelerometer sensor . The servers can be run on both windows and linux.

3. Network architecture

I. Overview

a. Network Structure

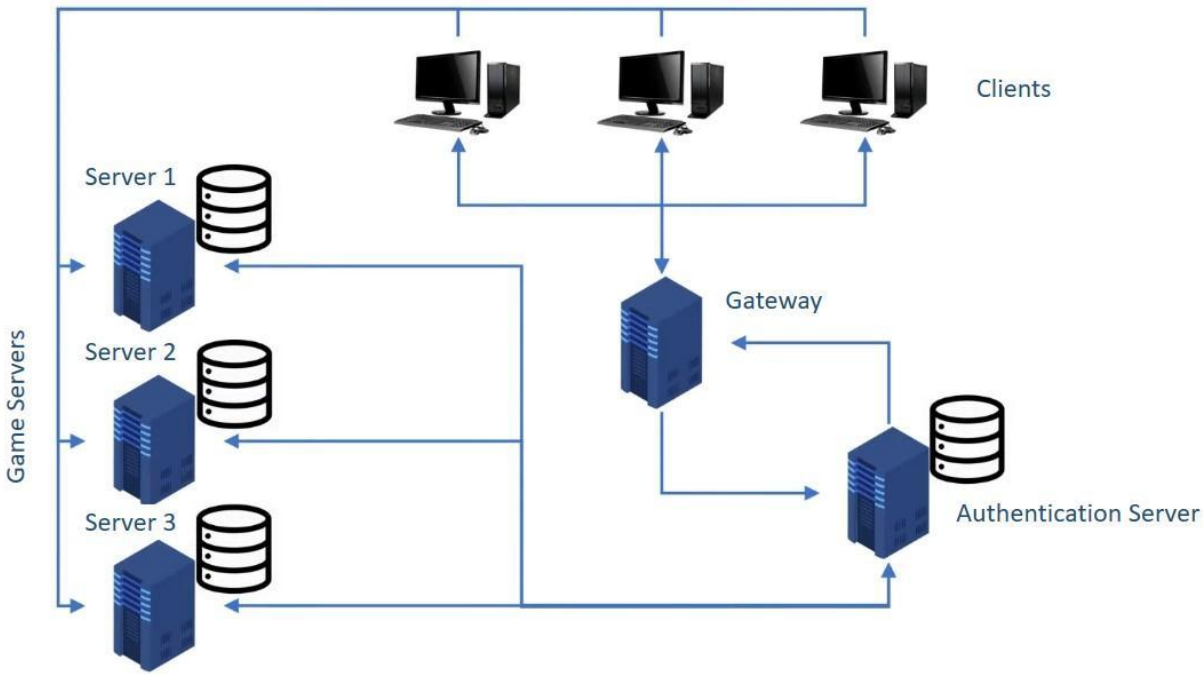


Figure 05 : Server structure

There are three server types:

1. The gateway server
2. The Authentication server
3. Game servers

The gateway server:

Connect player and the authentication server for authentication and secure the authentication server from unwanted traffic and isolating the authentication server from the public.

The authentication server:

Perform player authentication and issue authentication tokens.

The game server:

Obtain connecting players previous status through the connection made to the authentication server.

The game server will run the game world that holds players positions , players current status and synchronizes the players.

II. Security

a. User data security

FPSVR takes user data privacy seriously. User data is treated as private information and secured with following procedures.

- Username,Passwords are stored on the authentication server.
- encrypted with sha256 encryption scheme to avoid compromising user data on the server.
- Sha256 is implemented with (2^18) rounds (~1s for 1). Brute-Force attacks are prevented.
- Username,Passwords are sent between
- client-authentication server after 1 round of encryption.
- Passwords / usernames are deleted as soon as sent to the server for verification.

b. Token verification

On successful authentication auth server generate and distribute auth token to game server and the client. Client will then login to game server using this token. Player validation is done in the game server and direct connection is formed.

To avoid token reuse and other exploits, token lifetime of 30s is used.

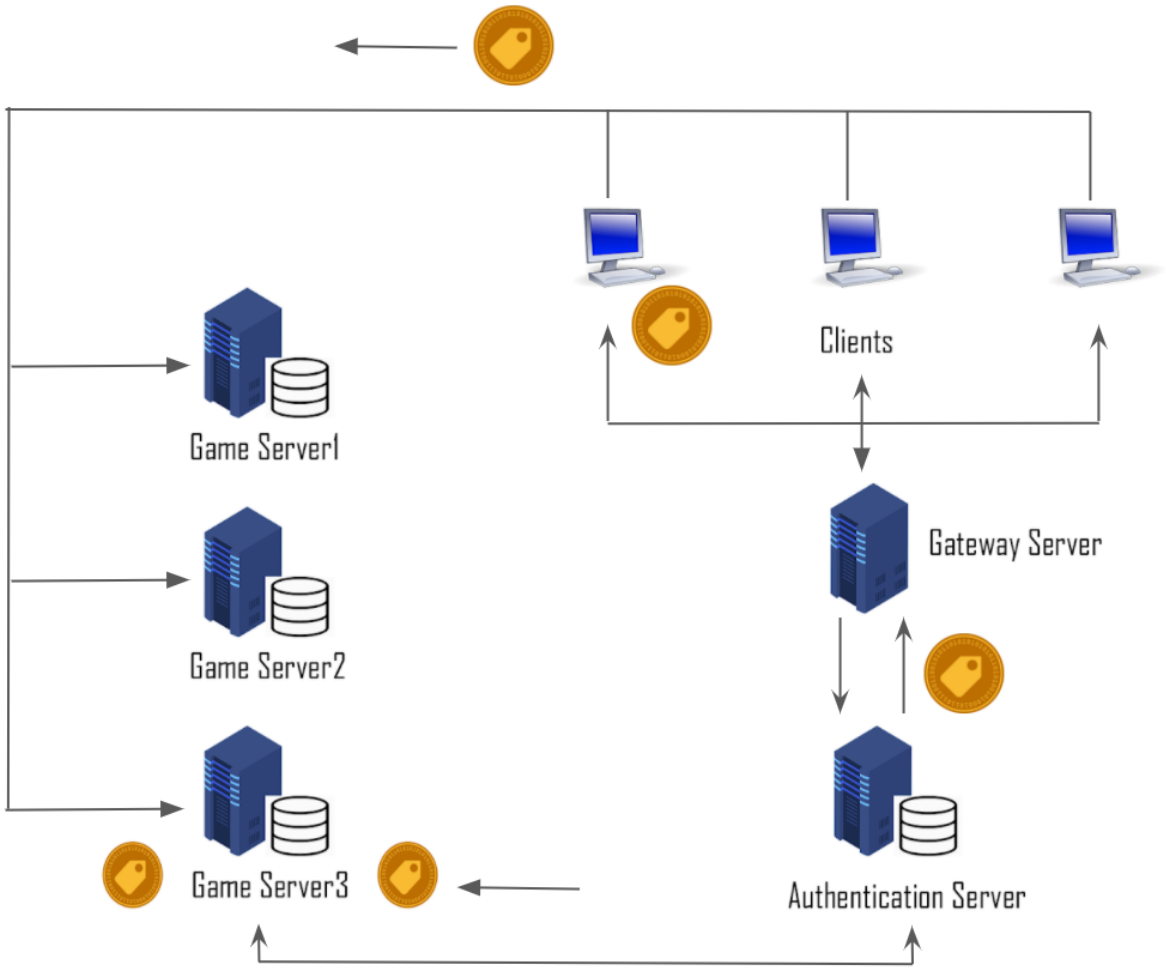


Figure 06 : Token verification process

c. Anti cheating mechanism

Player data is stored on the server. Player actions will be validated when an action is performed. Only if the valid world state on the game server is updated.

That way only valid player actions will be translated in the game and exploiting players can be identified and removed.

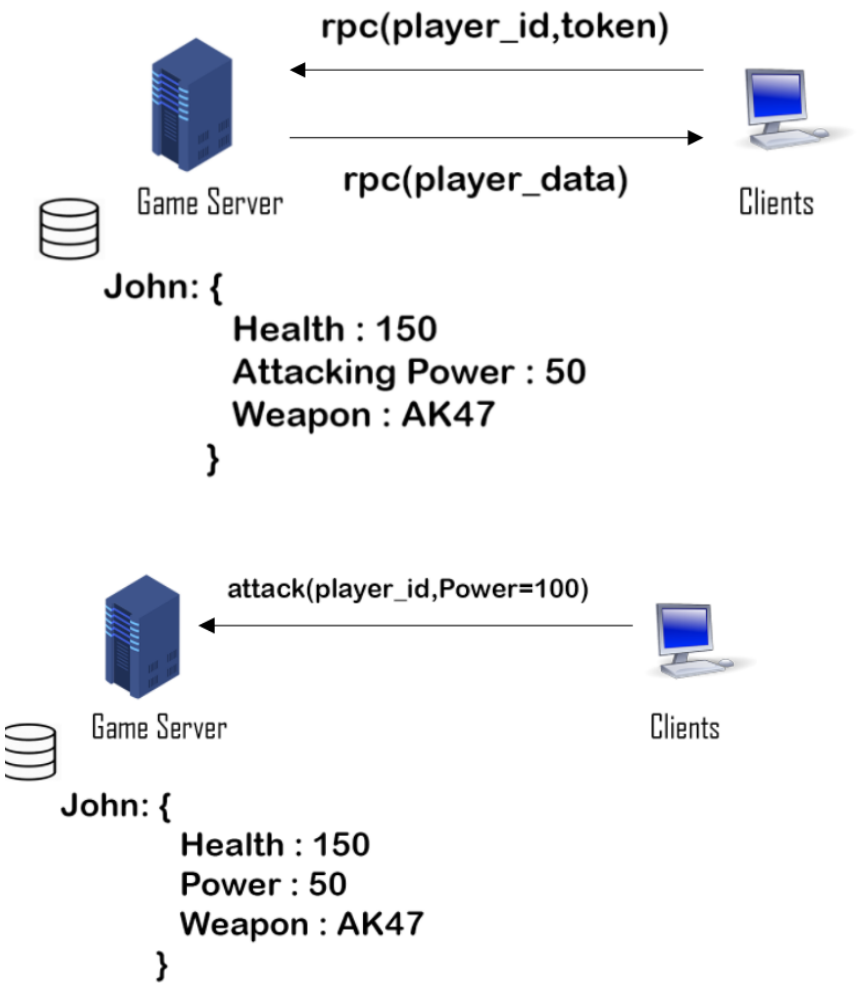


Figure 07 : Anti Cheating Mechanism

III. Performance

a. Solution overview

The game is supposed to run at a smooth frame rate while not using excessive network bandwidth and overwhelming game server performance.

Therefore, the client application is programmed to only update the game server on player position and other action data 20x per second. This approach reduces the network bandwidth and server load by around 3 times.

However, this forces the client application to guess the other 40 frames per second locally. Careful design of interpolation is used to solve this.

Also, since there is a small lag in any network, client game applications are made to run 100ms in the past at all times according to industry standard.

b. Interpolation

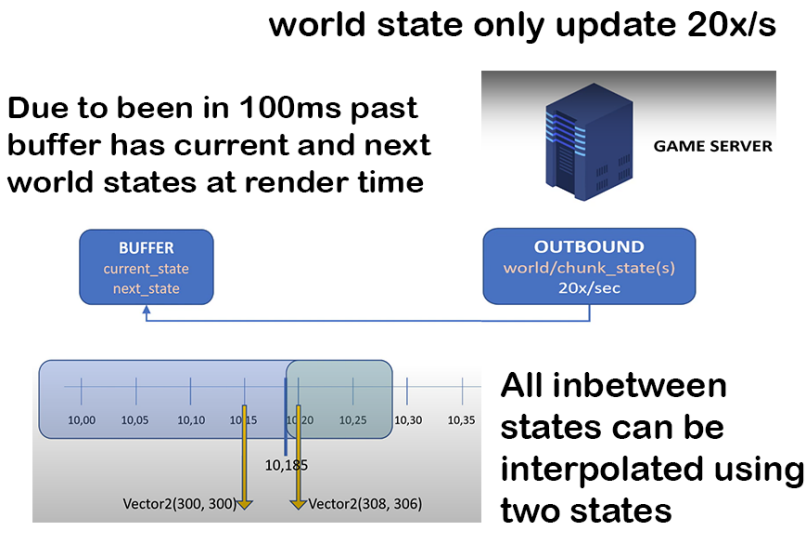


Figure 08 : Interpolation

Since the game runs 100ms in the past,under normal network operation the client application receives future states to interpolate the next state (positions,actions etc.) in 60fps rendering.

To render the next state ii interpolate the next two states that are received and stored on the buffer.

Buffer (simplified) : [distant past,near past,present,near future,distant future]

c. Extrapolation

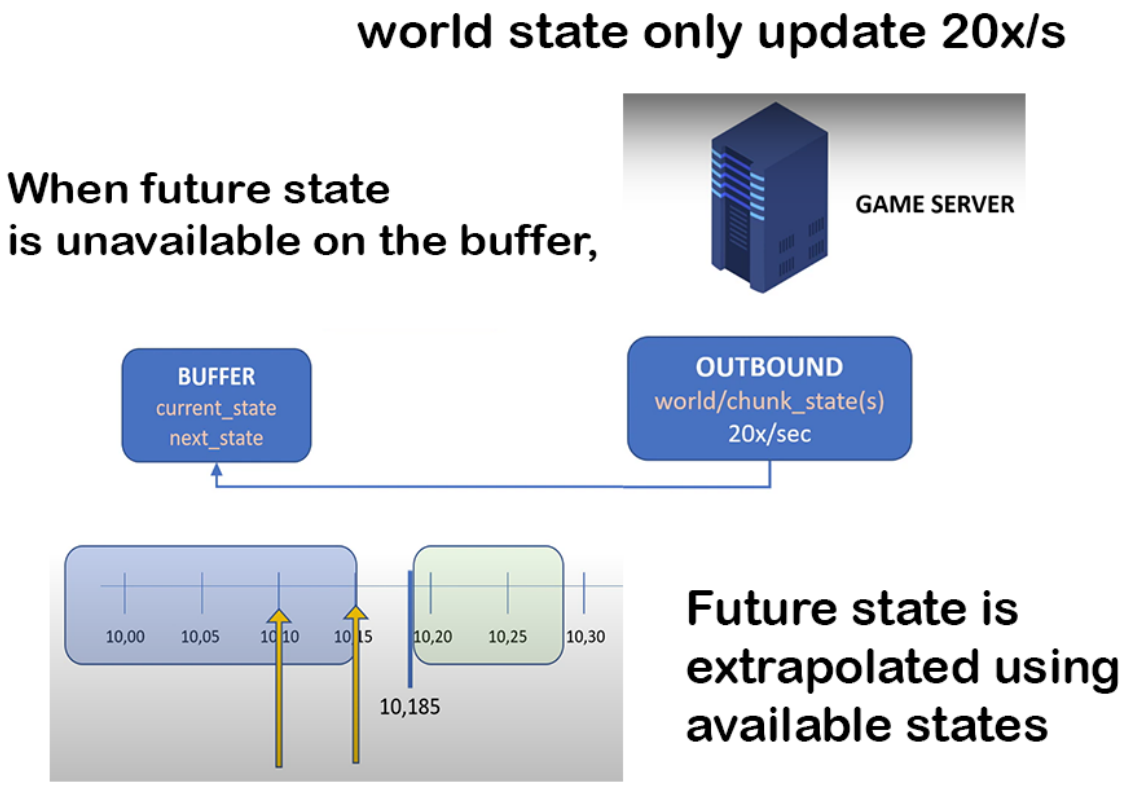


Figure 09 : Extrapolation

To render the next state when there are no next states in the buffer, client applications extrapolate the last two states that are received and stored on the buffer.

When the next state is relieved, it checks and corrects states if necessary. On large delays, this fails and the game lags.

d. Clock synchronization

Client application clock must be in sync with the server clock for correct data updates between them. A sophisticated synchronization mechanism which works regardless of small network delays is implemented to solve this issue.

IV. Cloud (AWS) implementation

a. AWS choice

AWS provides Infrastructure as a Service, pay-as-you go services such as networking ,storage and virtualization that are ideals for this project. Since we don't have concerns about the hardware we can focus only on the software part by using aws services. With aws regions and availability zones , the game backend becomes more closer and reliable to players in case of latency issues.

b. Implementation

By using simple cloudformation template aws ec2-instances , internet gateway , public route table, public subnets are created and configured for the backend of the game.

- Log in to aws account. Go to Cloudformation.

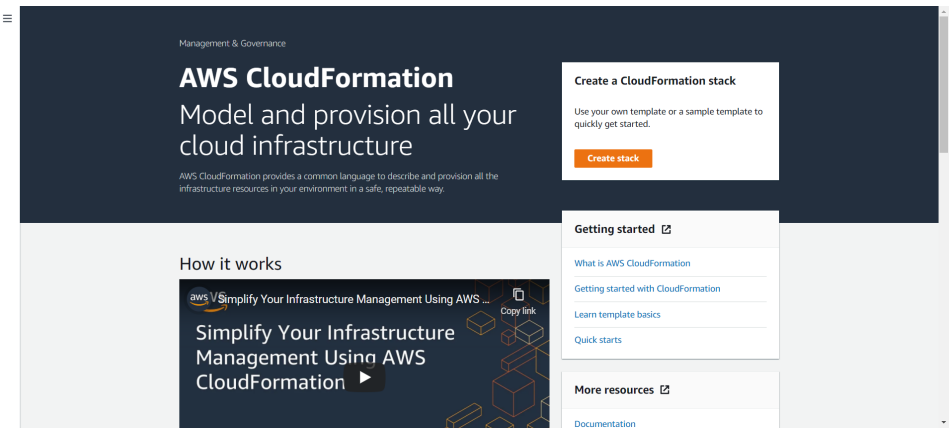


Figure 10 : aws cloudformation interface

- Click Create stack.

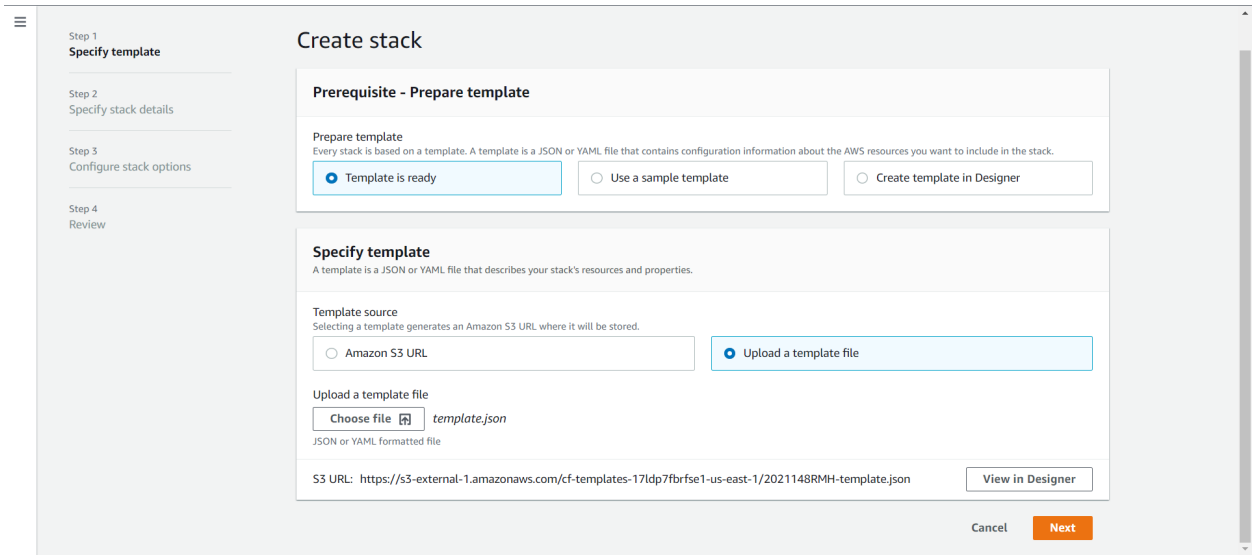


Figure 11

- Upload the template.json file.
- Click the Next button.

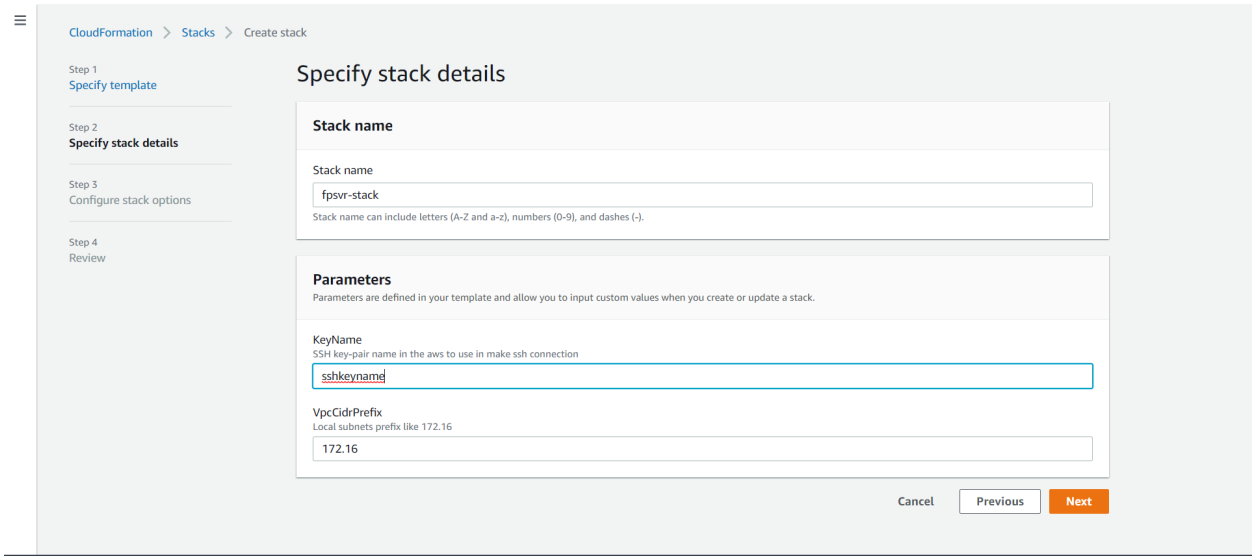


Figure 12

- Fill stack name , KeyName with a previous created existing ssh keypair name in aws KeyPair and valid CIDR prefix and click next button.

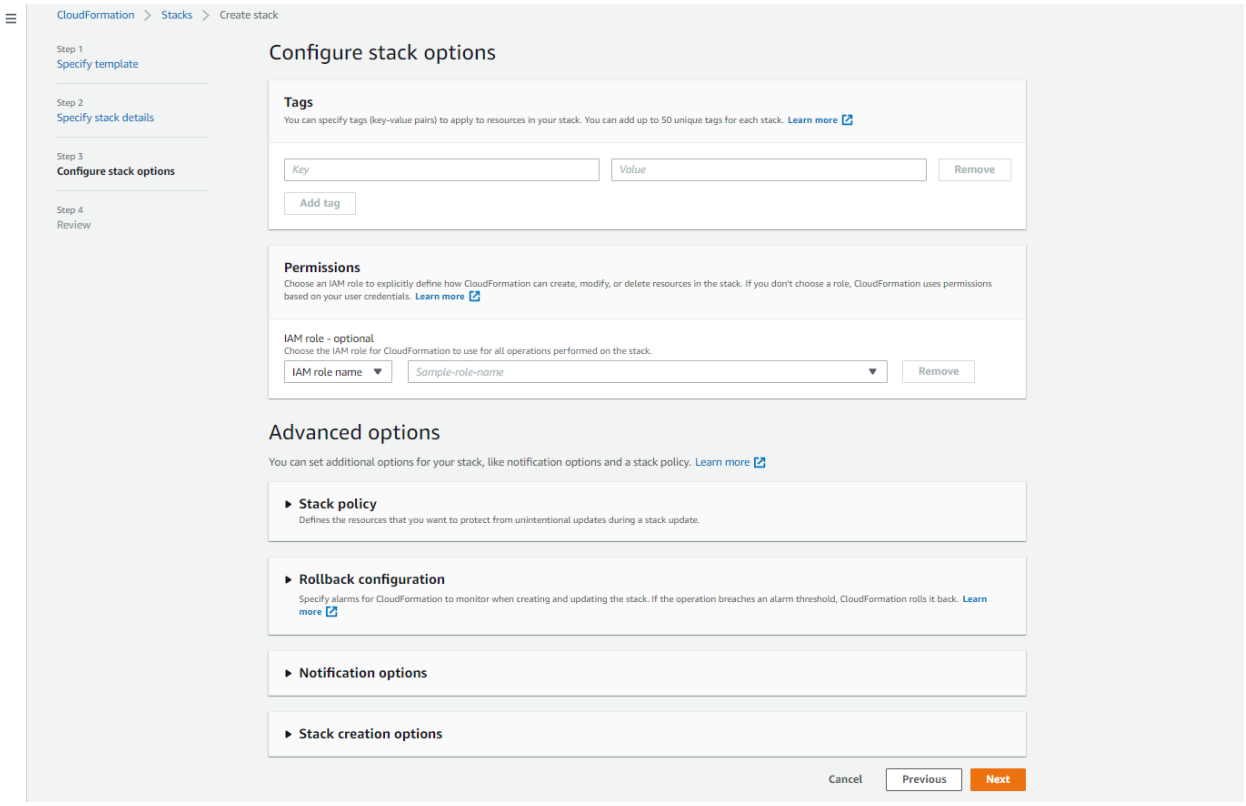


Figure 13

- Click on the Next button.

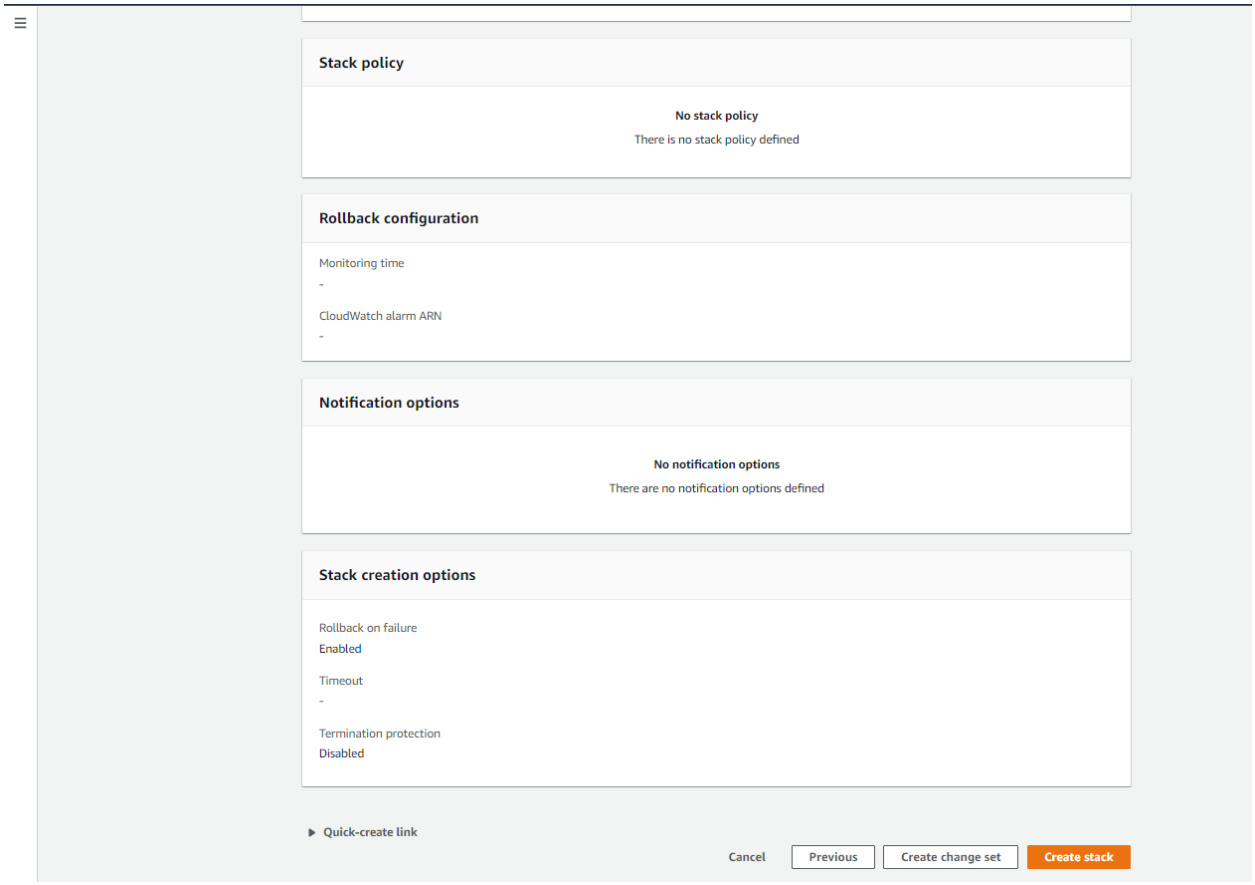


Figure 14

- Finally , check again the parameters are correctly provided and click create stack.

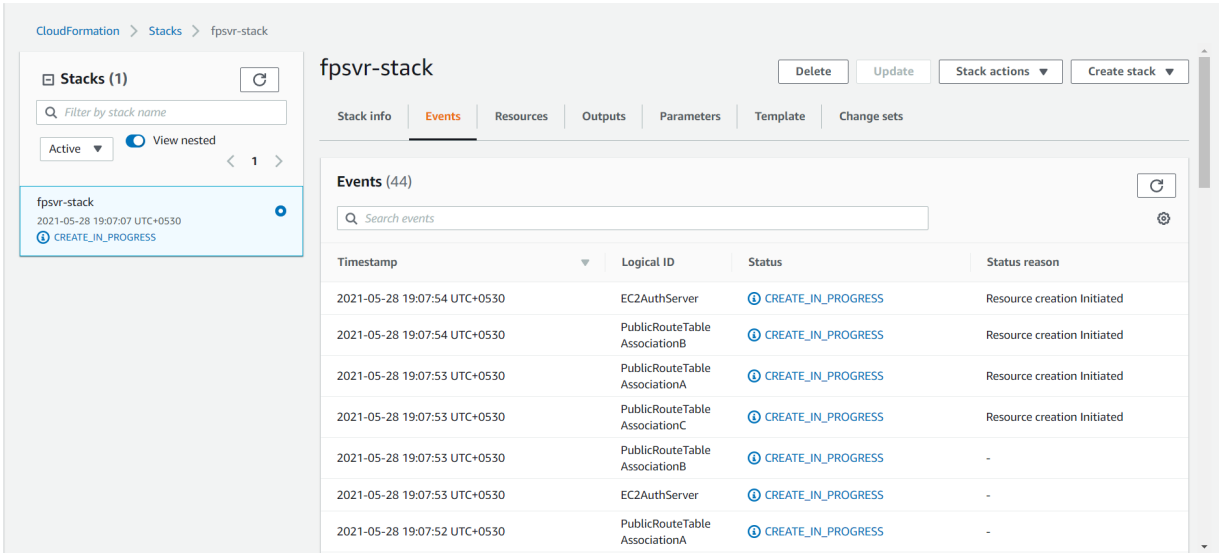


Figure 15

- Through the Events tab you can see the stack creation events.

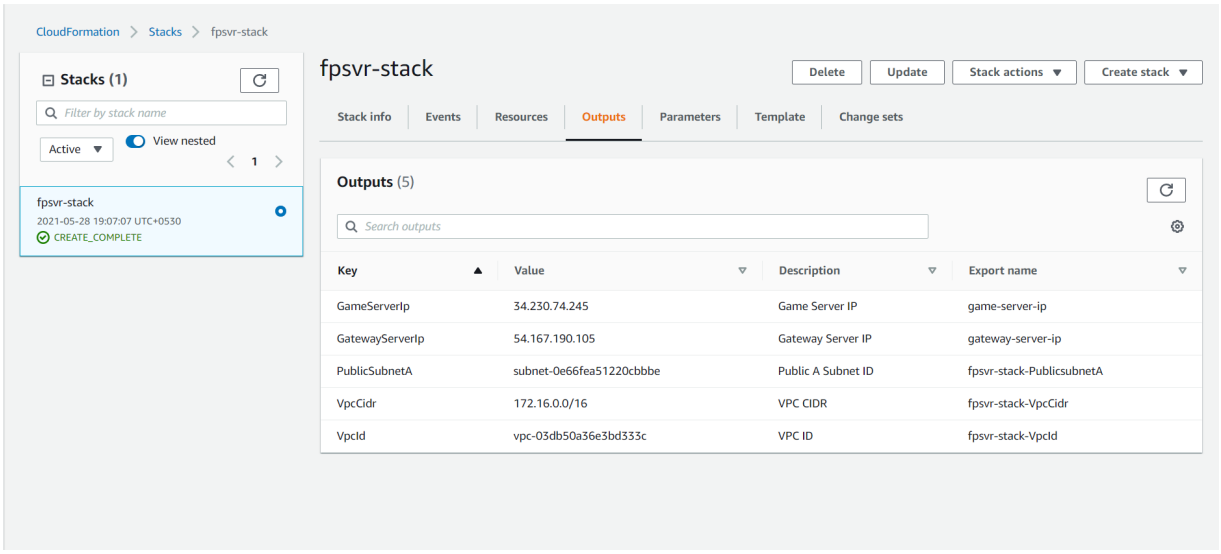


Figure 16

Once the stack creation is completed , you can obtain Game server and Gateway server public address through Outputs tab.

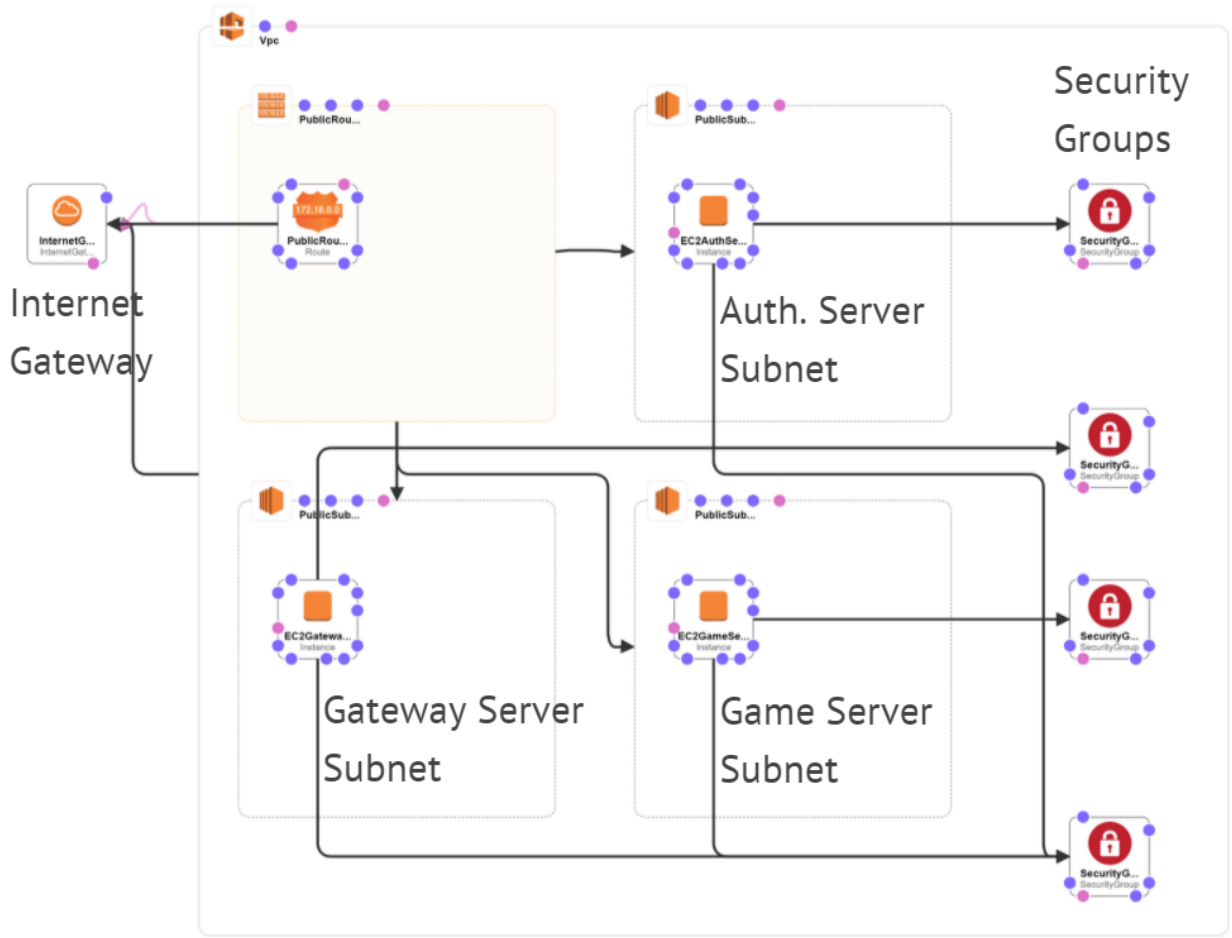


Figure 16 : overview of the stack design

3. Hardware Design

I. Overview

a. Sensor movement mapping

The sensor movement of tilting is mapped to roll and pitch values and then translated to movements in the game application.

```
// Calculate Roll and Pitch (rotation around X-axis, rotation around Y-axis)
roll = atan(Y_out / sqrt(pow(X_out, 2) + pow(Z_out, 2))) * 180 / PI;
pitch = atan(-1 * X_out / sqrt(pow(Y_out, 2) + pow(Z_out, 2))) * 180 / PI;

// Low-pass filter
rollF = 0.94 * rollF + 0.06 * roll;
pitchF = 0.94 * pitchF + 0.06 * pitch;
```

b. Actions

Controller Actions

Button press	:	Fire
Tilt forward	:	walk/run forward
Tilt backward	:	walk/run backward
Tilt left	:	walk/run left
Tilt right	:	walk/run right

Combined actions such as Tilt forward and left will result in combined game actions such as run forward and left.

c. Game Controller (Wired)

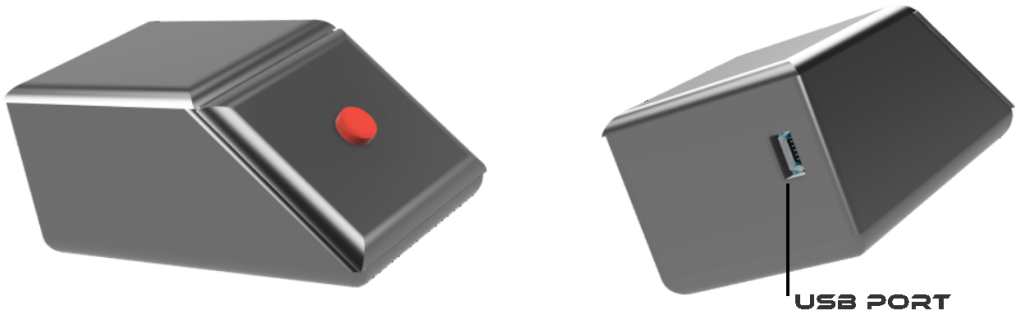


Figure 17 : Wired Controller (Front/Back)

d. Game Controller (Bluetooth)

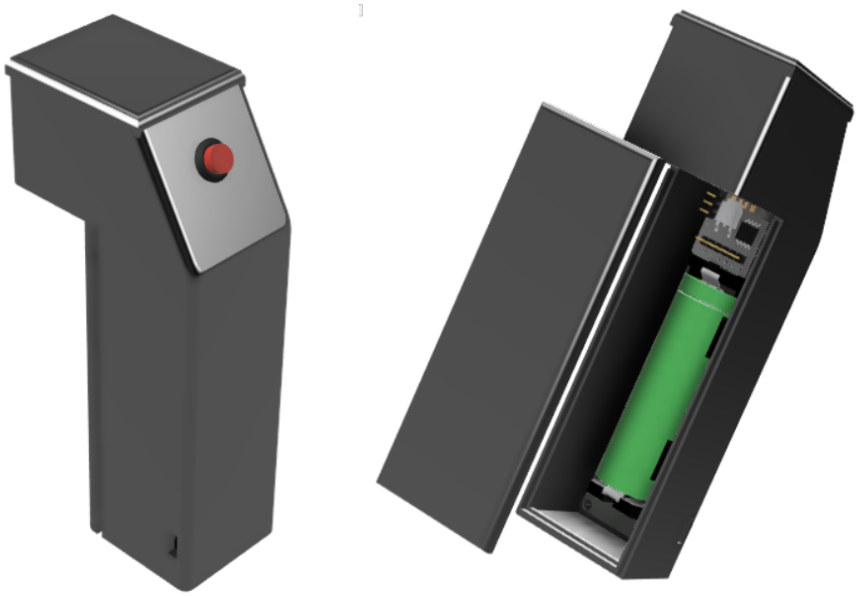


Figure 18 : Bluetooth Controller (Front/Back)

e. VR headset

The VR headset holds a smartphone screen as the VR display of the graphics.

By rotating the lens adjuster player should be able to focus an accurate single vision of the game. Rotate the wheel back and forth until two images on two eyes overlap each other and create a single view. Headstrps will be provided to wear it comfortably.

Smartphone should be securely placed with the help of fit adjusters.

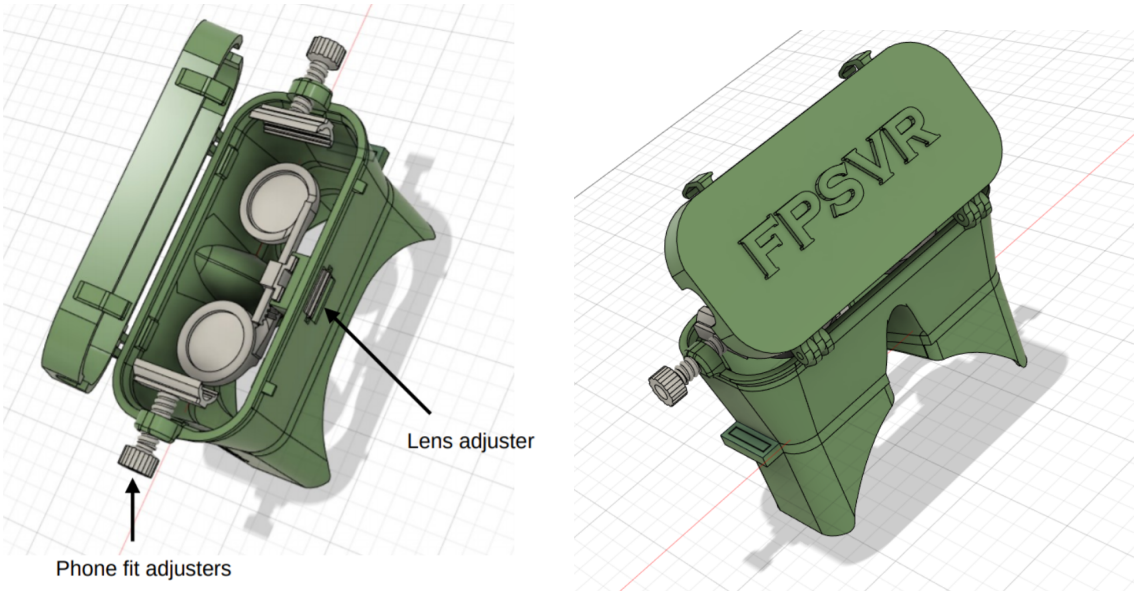
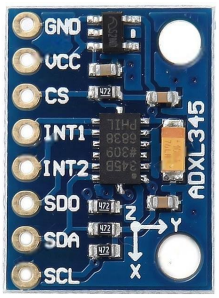


Figure 19 : VR headset Open / Close

II. Components

a. ADXL345 Sensor



- Low power 3-axis accelerometer
- High resolution (13 bit) measurement
- I2C digital interface
- Power consumption scales automatically

Figure 20

- Change in capacitance of structure on each axis is converted to an output voltage proportional to the acceleration on that axis.

b. ATTiny85V board

Figure 21

- Reads a single sensor, can be done with **smaller** and cheaper ATtiny85.
- Low power CMOS 8-bit microcontroller
- Based on RISC architecture
- Handles the data from accelerometer
- communication between main unit using bluetooth module

c. Arduino nano board

Figure 22

Used in the wired implementation as power consumption is not an issue on it.

d. HC05 Bluetooth module

Figure 23

- Bluetooth serial port protocol module
- 3Mbps Modulation
- Typical -80 dBm sensitivity
- https://github.com/jdunmire/HC05

e. Regular PushButton

Figure 24

f. 18650 Li-Ion battery + battery shield

Figure 25

- Easy and efficient solution
- More than enough power
- Easy charging with micro USB

III. Circuit Diagram Sensor model

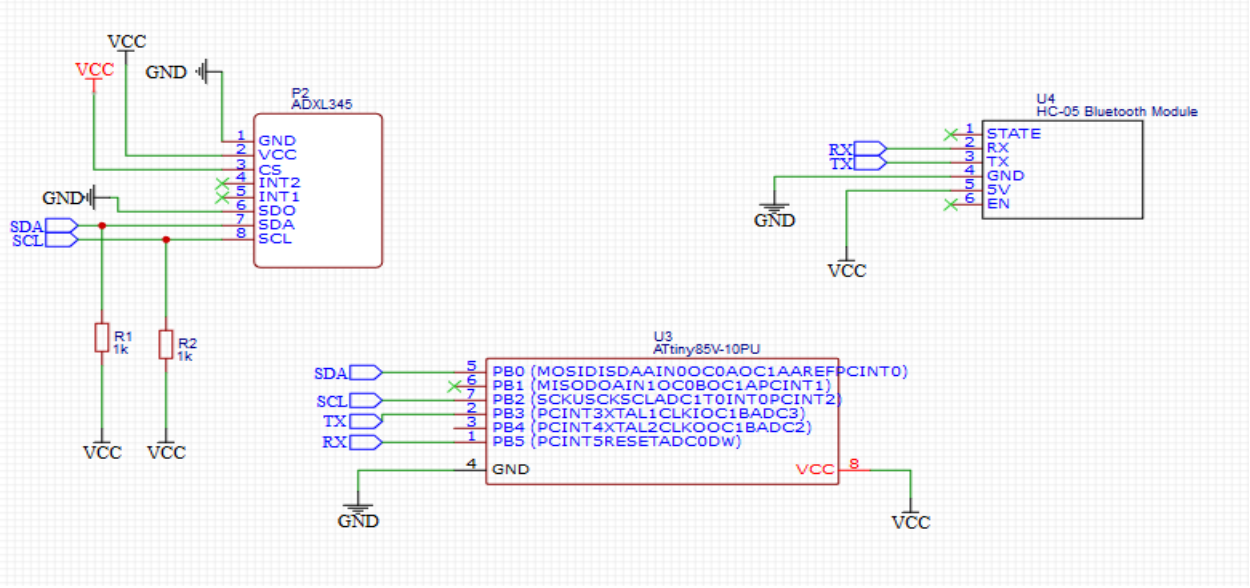


Figure 26 : circuit diagram

5. Testing

I. Sensor Calibration

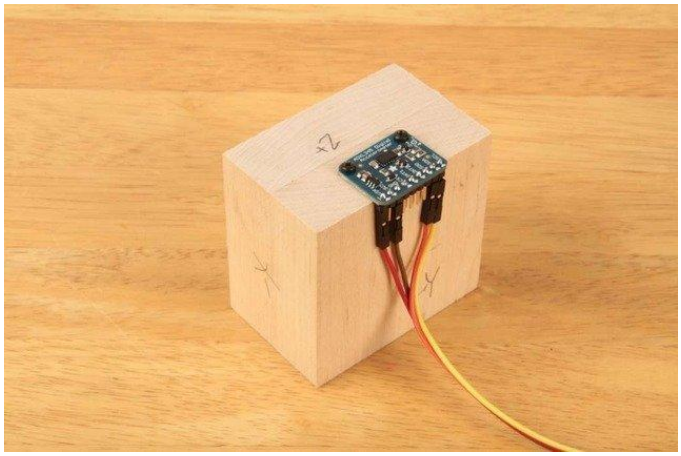


Figure 27 : calibration setup

The adxl 345 is a 3-axis accelerometer which can measure both static and dynamic forces of acceleration. The earth gravitational force is a static force, which should result in a sensor reading of 1g.

Therefore to calibrate the sensor is placed exactly flat on a surface and measure the output (use serial monitor to monitor output). Then, required values are written to 3 offset calibration registers to get x,y,z outputs of 0,0,1g.

II. VR module Testing

Implementation of VR is tested with gameplay by running the game on a normal smartphone (Huawei GR5 with Android 9).

Google VR services are used to implement VR play on the smartphone. Therefore, no access to the smartphone sensors were required.

Google cardboard and Google VR services directly communicate with the godot openVR module. Therefore, no calibration or hardware programming needed to play VR content on the screen.

III. Movement Smoothing

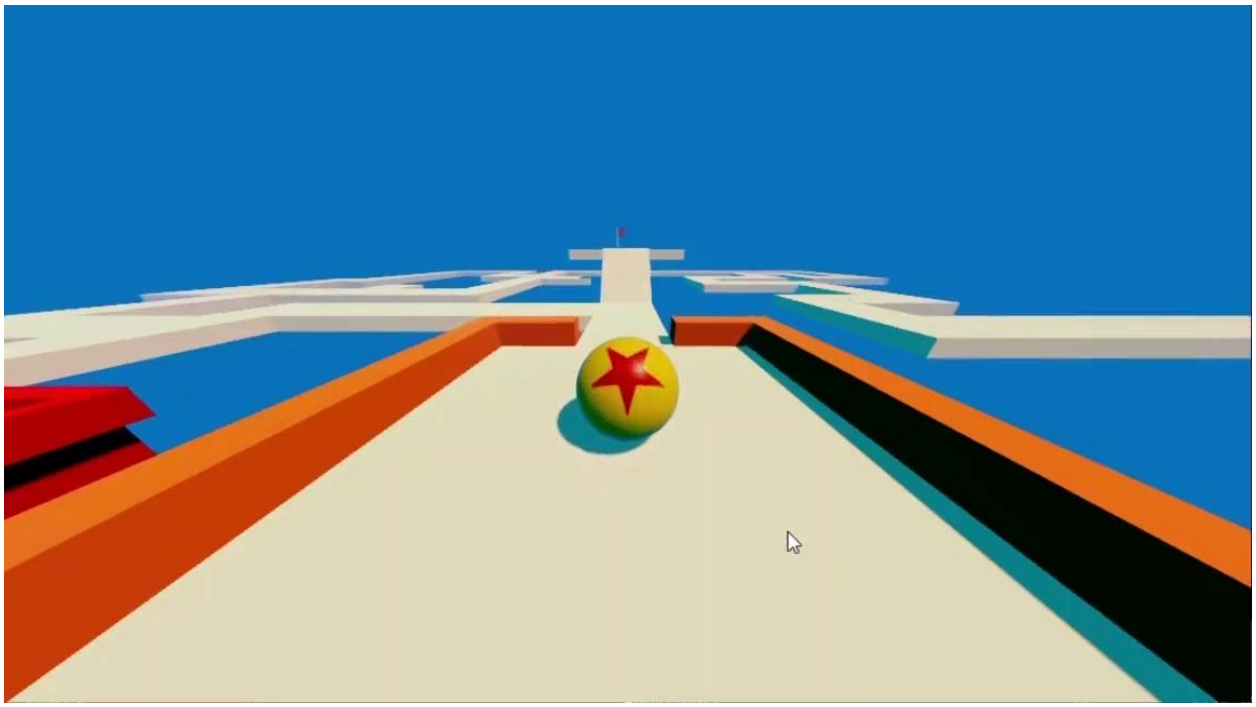


Figure 28 : Test environment

Sensor output smoothness, reliability and speed was tested on the above testing environment.

Fine tuning was done on the fly according to the behaviour of the movement from sensor input.

IV. Network Testing

Since the game was built in a way that the servers will automatically start at the beginning , the current test frameworks can not be applied to test the network in the game . Therefore ad hoc testing methods were used to do the networking test .

Test cases and reasons:

- Terminate the server while playing the game to identify each units' behaviour when an external error occurred.
- Disconnect the player from the network to identify and calibrate the time taken to identify the connection lost.
- Play game while running apps that consume high bandwidth to check performance of lag fixing issues.

Terminate servers while playing gam

---END---