21 June 2019 06:17 PM

FPS GAME ENGINE

Name:	Debdip Banerjee
Date	18 June - 21 June ,2019

Contents:

About

21 June 2019 06:22 PM

This Game Engine Is inspired By Wolfenstein 3d(1992) Game Engine which was made By John Carmack of Id Software and been used in Catacomb 3d & Spear Of Destiny.

In the Process of Making this Engine I've learnt many advanced Graphics Topics in Depth.

Like,

1. Raycasting
2. Collison Detection
3. Field of View limited on Single Plane
4. Movement in 2.5D Space

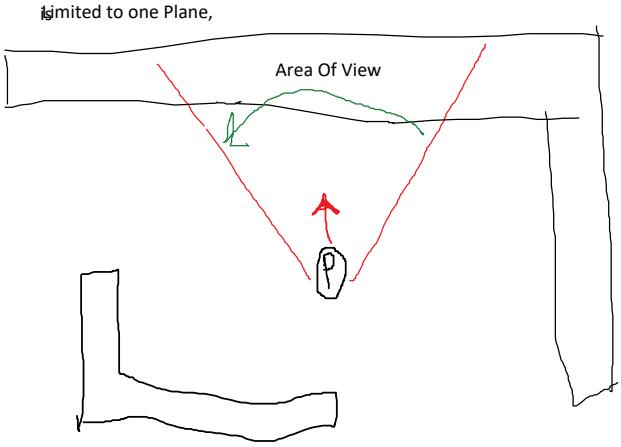
Please Open The Link Below:

https://en.wikipedia.org/wiki/Wolfenstein_3D#/media/File:Simple_raycasting_with_fisheye_correction.gif

Fps Engine

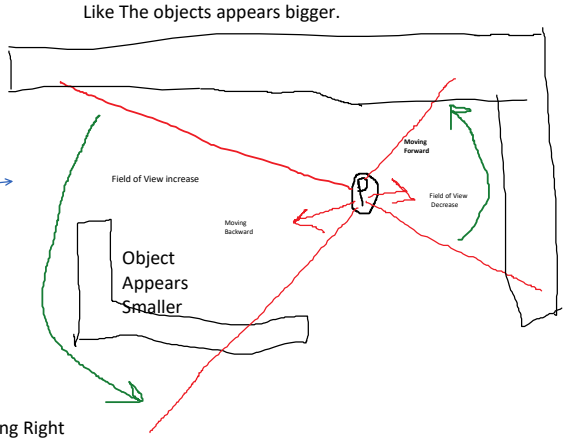
19 June 2019 09:48 AM

In this Game Engine The field Of view is limited to one Plane,



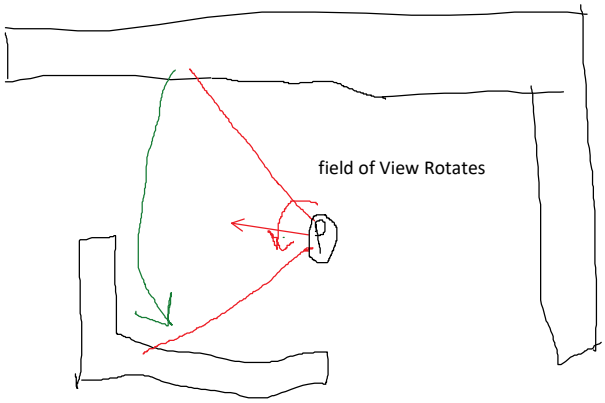
Forward/Backward

and when player move forward and Back ward
Field of View increase or Decrease,
Like The objects appears bigger.



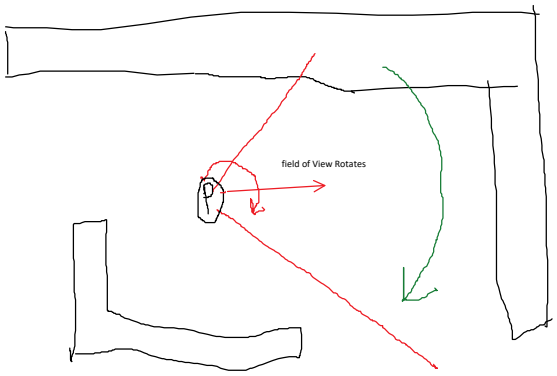
Object Appears Bigger

Moving Left



Moving Right

So, when player moves
Right or left field of View Rotates



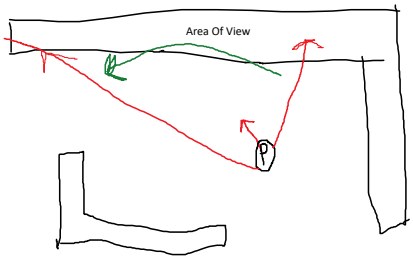
Command Prompt Console

19 June 2019 11:26 AM

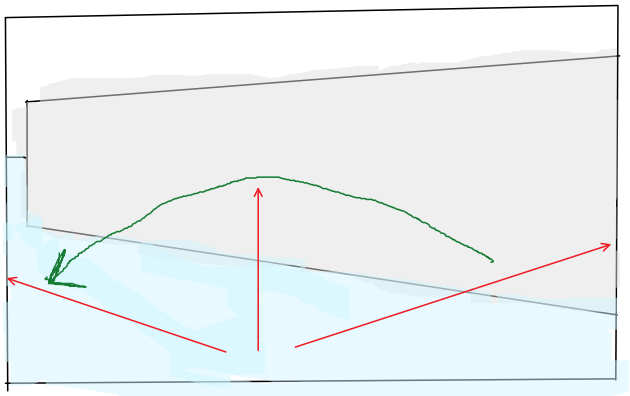
Due To Lower Speed of Std::Cout
We are using Screen Buffer

So Basically what we're Doing, we are making a Grid with walls and a player at certain position.
And we're limiting the Field of view at a single plane.

To Implement something like this we are using Raycasting technique.

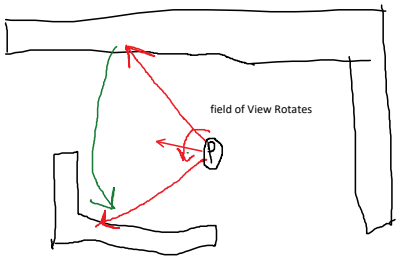


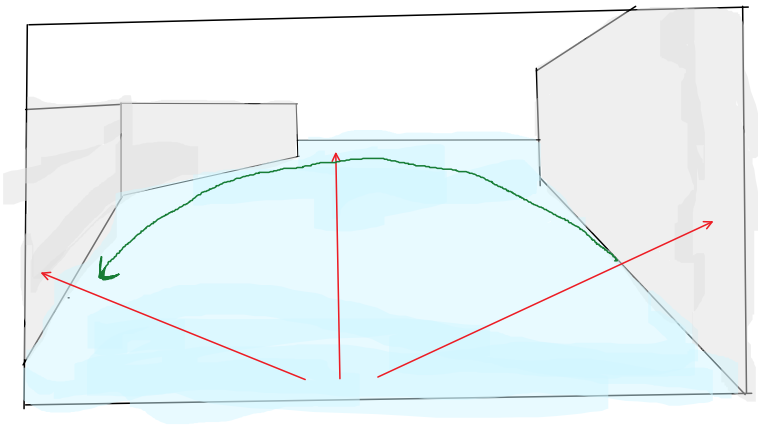
Now this Top Grid View
In FPS view Becomes



Moving Left

View Rotates

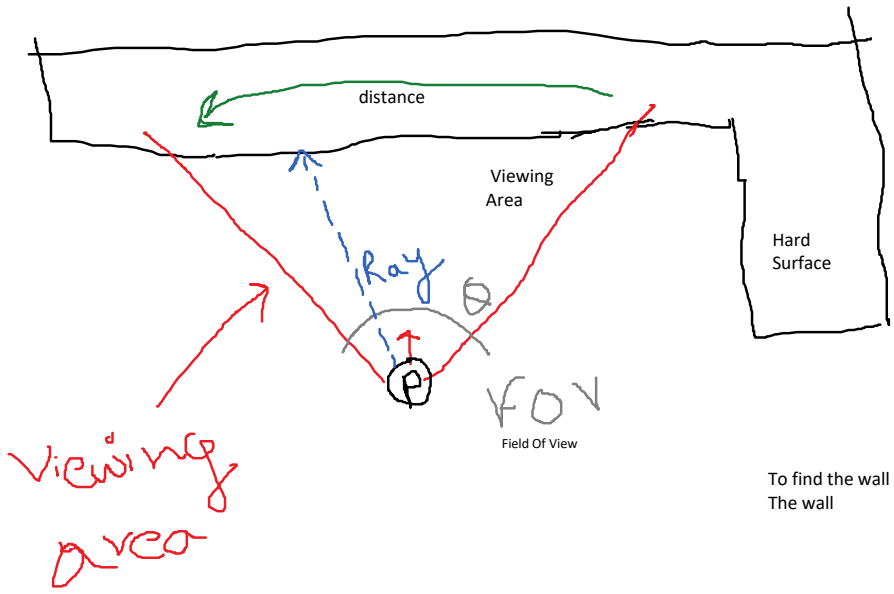




We are using Ray casting To find the Distance Between every objetcts
& then Generating objects Based on it,

Raycasting

19 June 2019 12:17 AM

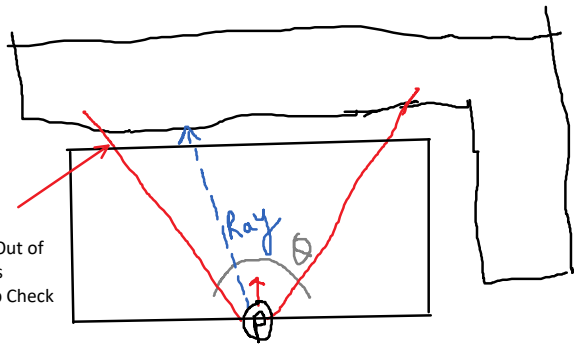


Raycasting:
Shooting Rays from
Point 'p' and see how
Far it goes until it hits
The surface.
We will hit ScreenWidth =120
Rays to find the distances,
Which will give us Array of distances.

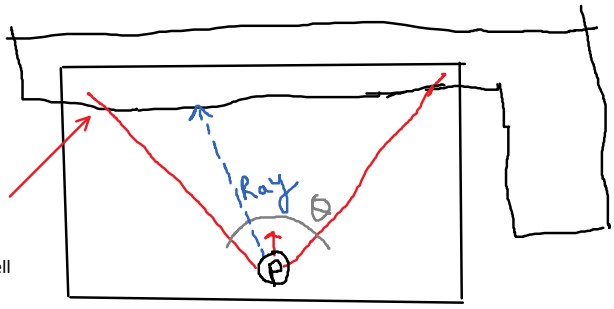
To find the wall we keep incrementing the ray and check until it hits
The wall

If Ray is Out of Bounds
Ray hits a wall and stop the ray casting

When Ray is in bounds
We need to chcek where ray hits its Wall cell (#) or not



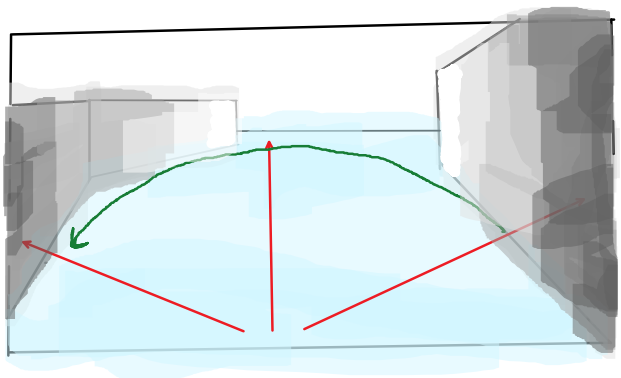
Ray is Out of
Bounds
So stop Check



Inbounds
Ray hits
It's a Wall cell

NOW we see the perspective,

When distance to wall
Gets higher, celling gets bigger



We can see more amount of floor & Celling
When its far
This gives us perspective

So we need to find the distance from the celling & floor

As we've implemented the wall, celling & floor
We have implemented controls to move

Now Need to implement Perspective of elements

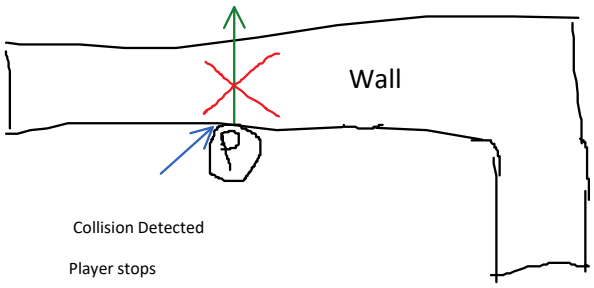
To Implement the Shades we are using
ASCII extended list

Collison Detection

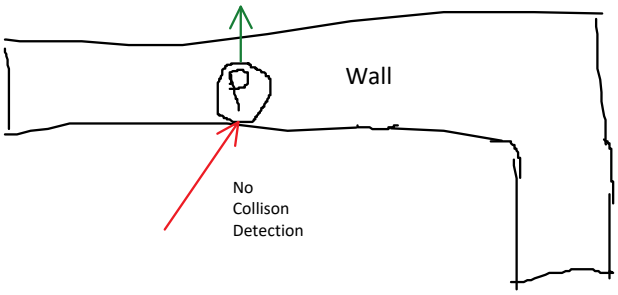
21 June 2019 12:55 PM

We are now implementing Collison detection
So when the player hits a wall it stops instead going through it

With Collison Detection:



Without Collison Detection:



Now we are trying to detect the corners of Blocks

Ray from Corner to player

Ray from Player to Corner