Jack Storm
Primitive Pac Man Log

6th November, 2014 – Evening

Decided to make the game board using cubes and a plane. The plane is the floor and the walls are made of the cubes. The walls will be a child of the floor so it will stick with the board if it needs to be moved or scaled. The player is represented by a cube.

Player movement will be determined by having a player constantly move forward until it hits a wall. Only then will it stop. If the player presses a direction, that direction is queued up. Each update, it will check if it is able to move in the queued direction. If it is able to move in the queued direction, current direction will be set to the queued direction.

Problem 1: How to handle the queued input.
Example: If the player is moving to the right and up is pressed, the player will keep moving right until it is able to move up. Bounced ideas off of a few friends to see what they thought. I hope that is alright, I like to bounce ideas of people to make sure that they are feasible.

> Possible Solution 1: Create trigger boxes at each of the intersections that allows for the player to change to the direction that they had queued. While this would work, it seems very hacky. There is probably a better way to do it.
>
> Possible Solution 2: Ray-cast off the left and right sides of the player. Check if the ray collides with a wall, if it does, then the player cannot turn.

Problem 2: When the player hits the wall, it has the tendency of bouncing in a weird direction. Sometimes it will unaligned itself from the grid.

> Solution: Set the constraints of the players rigidbody to freeze the rotation and position. This keeps the player on the correct plane. While this keeps the player on the correct plane and snapped to four directions, it got rid of the default collision response. Solved this issue by writing a script that sets the players direction to none with OnCollisionEnter. In the update, if the current direction is equal to none, it will not move.

When the player runs into the wall the player will stop moving until an input key is pressed. However, the player is stuck partially in the wall so if forward was pressed again, then the player can phase through the wall before it checks for collision again. Right now, the wall handles the collision with the player. It sets the players movement direction to none and it pushes it back so it is not in the wall anymore. This lowers the chance of the player phasing through the wall. However, it does have the player looking like it bounces off the wall, even if slightly. I will come back to this if need be.

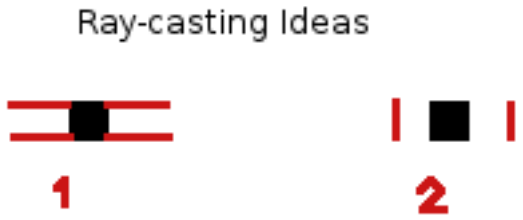
7th November, 2014

Added some simple textures to the objects to help differentiate them. Other than that, I didn't have much time today.

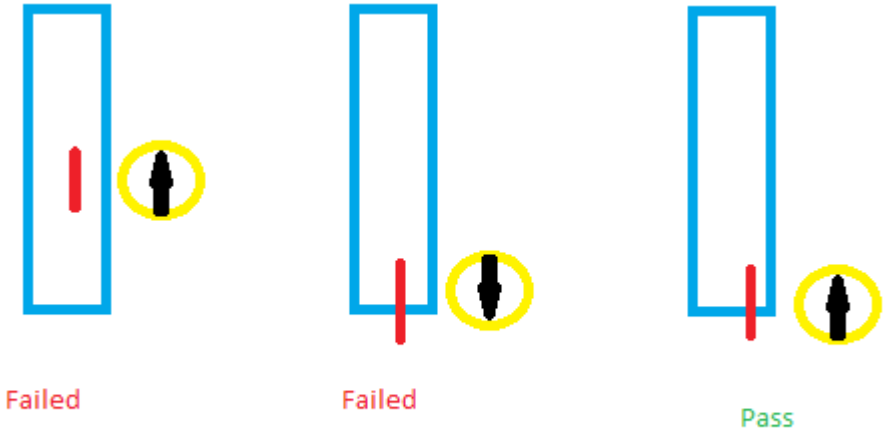
8th November, 2014

Did some more research on using ray-casting. Came up with two ideas and how to go about it.

Originally I was going to go for idea 1, but I want to try implementing Idea 2. So instead doing for ray-casts off of the sides of the player at the corners, I will just be doing two ray-casts going parallel along the sides of the player, and hopefully that will yield the same results.



I think I misunderstood how the rays work. It seems it will only register as a hit if it is going from the outside to the inside of the wall. So chances are, idea 2 won't work based on that assumption. I might be missing something but for now I will just work on getting idea one implemented.



Using idea 1, there will be two rays on a side. Both those rays need to not intersect for the player to be able to turn in that direction. Using idea 1, I was able to implement the queued input and set up the turning for the player. After some testing, realized that if the wall was smaller than the player then the player still had a chance to turn in to the wall. Added another ray that comes out of the middle of the player and goes to both sides.

Decided to have the camera follow the player from above with a certain yOffset, but player is always in the middle of the screen. Now that the input queue is done, I am going to finish building the rest of the level.

Next thing to implement in terms of player movement is the warp around points. Since the map is symmetrical along the z-axis, I can set up a trigger at the warp points and just negate the players x value. Got the player to warp to the opposite side of the map. Modified the camera so it will not follow the player on the x-axis. The warping made the camera movement jerk when it followed the x axis.

Made a mistake. You can't see all of the play area normally. Didn’t realize that I was in free aspect mode when testing and that everything fit nicely on the screen. Moved the camera out and put it at an angle instead of being top down. Seems to work well enough.

Originally I had the warps set up int a way that the play would ray-cast forward a little bit and when it hit a warp the player would be transported to the other side of the play field. I want to try using a trigger instead to see if I can work something else out.

Successfully switch the warp from ray-casting to using triggers. Honesty don't know why I did that but it was fun.

The board is made up of blocks that act like walls. Most of them are 2x2, 4x2 and 6x2. For the most part, that seems to work. But there are some locations on the board where there are two 4x2s together making up an 8x2. Sometimes this set up causes an issue with the ray-casting for the input queuing, where the player thinks that it too close to the wall section to be able to turn.

I have messed around with increasing and decreasing the speed but I can't recreate the issue of being unable to turn. Not sure why I can't hit it again.

Working on writing a script to place pellets around the board. Got the pellets to spawn starting at the bottom left corner. It spawns a grid that is adjustable in size. After the pellets are spawned, it removes the ones that are spawned inside of anything with the tag 'Wall'. Had a little trouble with getting the pellets to be destroyed if spawned in the wrong spot. I eventually figured out that I was trying to delete the script component instead of the actual pellet Created a special prefab that acts as an anti-pellet-spawn block. I use it to get rid of pellets from the center, the corners and near the warps.

The player can also collect the pellets, but currently all that entails is pellets being destroyed when you hit them.

I am adding a sound effect into the game when the player collects the pellets. I want to make it so when you collect multiple pellets within a certain amount of time the pitch goes up. First attempt at this, I added the audio source to all of the pellets. I had a static variable in the script that was attached to each pellet so I could control the sound. On second thought, if I have the player just be both of the audio source and listener, I will only need one audio source. It will also be easier for me to track the time between each collected pellet and to adjust the pitch.

Now when the pellet gets collected, it signals the player to generate a pitch. Longer the player collects consecutive pellets, the higher the pitch gets until it is capped. If the player takes too long the pitch just goes back down to the default.

9th November, 2014

The issue where sometimes the player would get stuck on the wall and not be able to turn in a valid direction came back. I adjusted the front ray-cast on the sides so it wouldn't start as far out. I believe the ray-cast would sometimes start inside the walls if the player hit it head on, and at some intersections it would register as a hit, making the player be unable to turn. Moving the ray-casts closer to the player has hopefully removed the chances that the player will get stuck in.

Working on getting a simple AI to move around. The AI will share some characteristics with the player in terms of movement. So I have created a base movement class that has the basic directions, setting of directions and boundary checking. Both player and AI will inherit from this class. Both player and AI also make use of the queuedDirection.

I am working on a seeking ghost and a wandering ghost movements. I have got the wandering one completed so that it will stay on the board correctly and it will choose a new direction that it wants to go every few seconds.

Right now the seeking ghost will seek, but it ignores all walls. I would probably have been better off if I attached everything to a grid or a nav-mesh to find the path to move around properly. But I don't think I have enough time to put one in now. I think I can get the seeking ghost to try to follow along properly on the board instead of going through the walls as it seeks the player.

By using the same boundaries checking that I use for the Player and the wandering ghost, I was able to keep the seeking ghost out of the walls. During the update function of the seeking ghost, it checks forward, to the left and to the right. If it is a valid direction it checks the distance from that spot to the player. Which ever one is closest it will go that direction. Currently, although the ghost never checks backwards, it still does/can do 180's at intersections. I think this is because it checks every frame. So in this case, when the ghost comes to an intersection and the player is behind the ghost, the ghost will turn, and then turn again on the very next frame.