**20/03/17 (1 Hour Spent)**

Ok so I have an idea, I want to make a MiniGolf game within Unity! Where the heck do I start??

Aim low and functional, all the art and tweaks etc can come later, long as I get core gameplay and mechanics right.

**Thinking out loud**

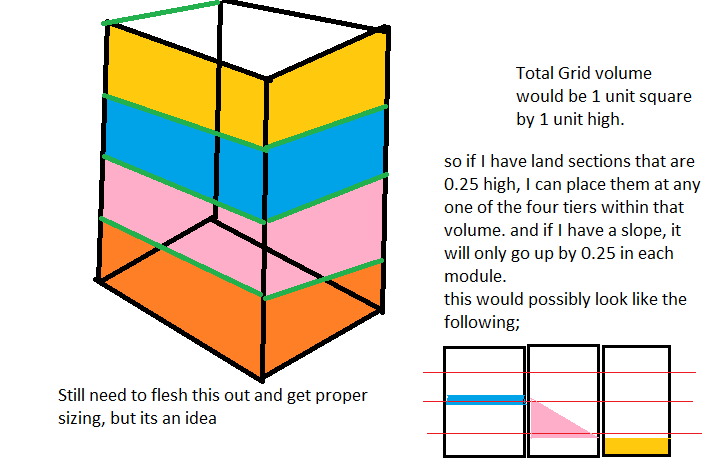
Firstly I was thinking that I want the course to be of a modular nature so that I can build levels relatively quick within the editor. So do I want it all to be on one level (flat) or do I want to put hills in it? Think I do want some hills to hit the ball up, as a totally flat course would be a bit boring.

So if I want it to be modular I will have to make each section easy to click together.

If I say each course ‘module’ will fit in the part of a grid so that they can be put in and be sure that they will fit together. That will definetly make it easier to construct.

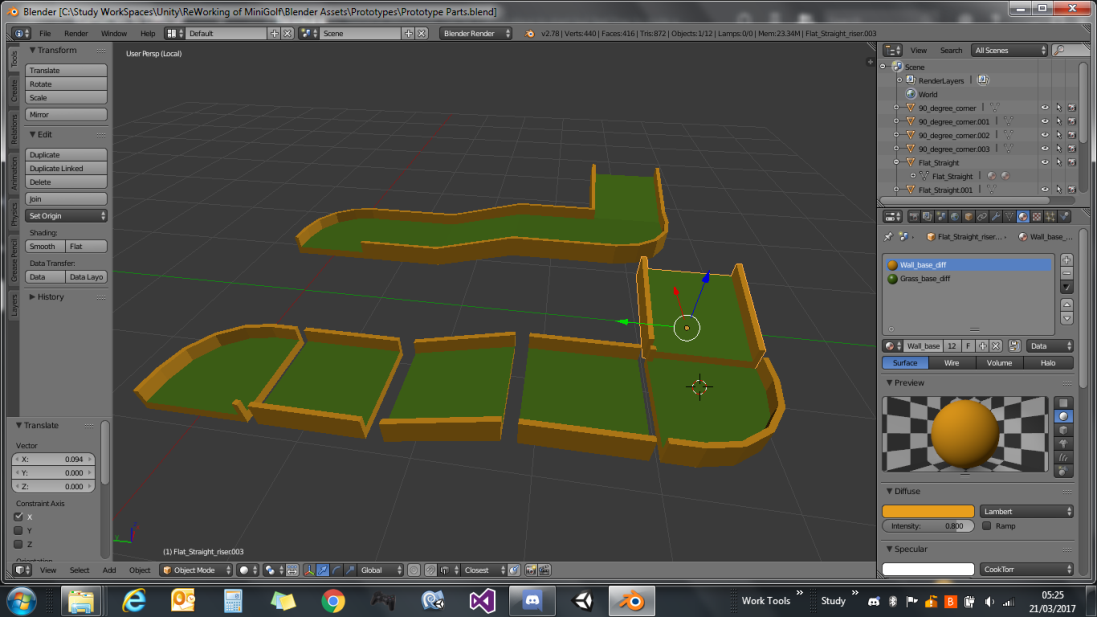
***Question****: how am I going to handle the ups and downs of it, like putting in slopes etc?*

Thought: well, what if I say that each of the grid positions could have 4 different heights to choose from to place the course part on, like the following



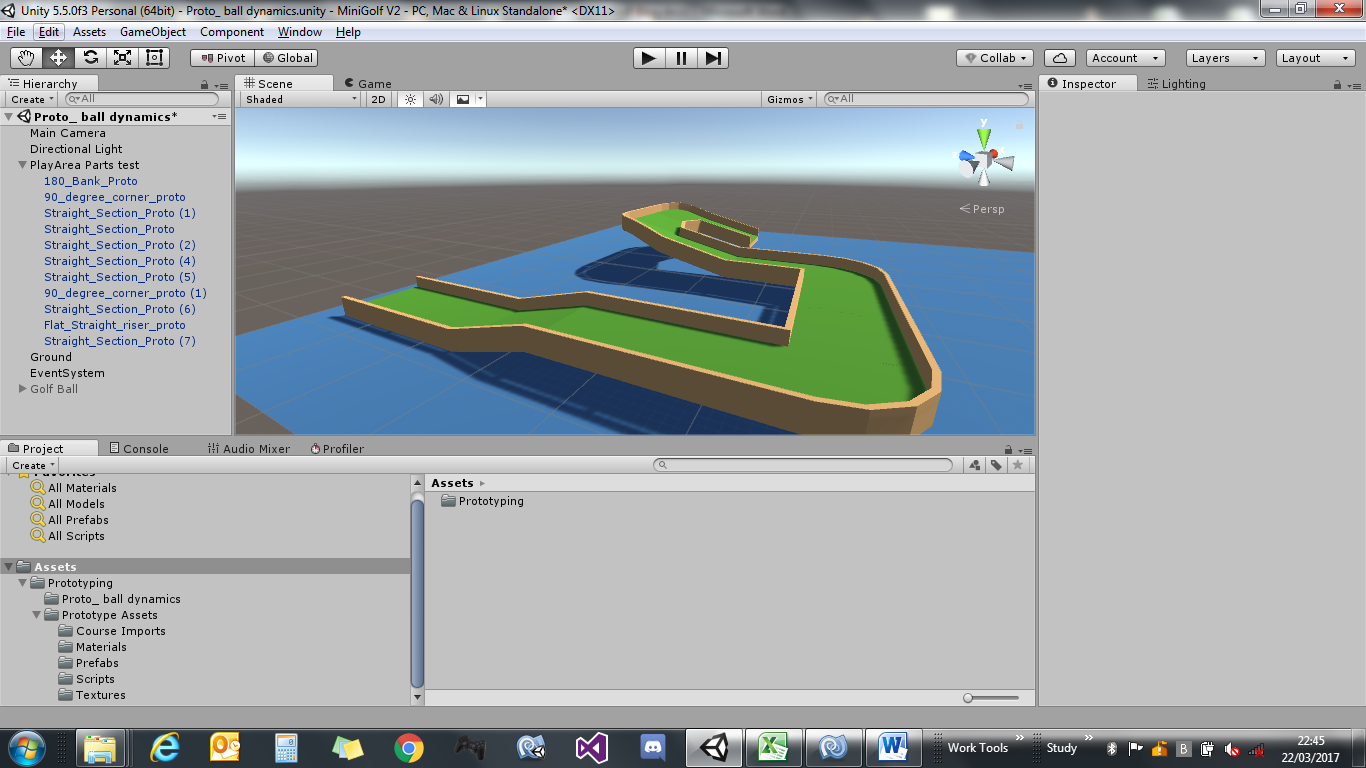
**21/03/17 (2 Hours)**

So ive made a couple of test pieces in blender, each 1x1x0.25, and ive set the proper transforms on them.



They seem to be going together half decently and all line up if they are 0.25 in height apart.

So ive exported these to FBX individually and put them into a Unity layout test scene to see how they looked, and added a mesh collider to each section and also prefabed them too as prototype parts.



Quite happy with that, I did notice that the import settings were a 1 to 1 scale.

I applied transforms in blender to 1, exported the FBX with a 1 scale. When importing to unity they have an absolute scale of 100, need to go back and see whats up with that. Think ive forgotten something. Still looks ok tho., so that will do me for tonight I think.

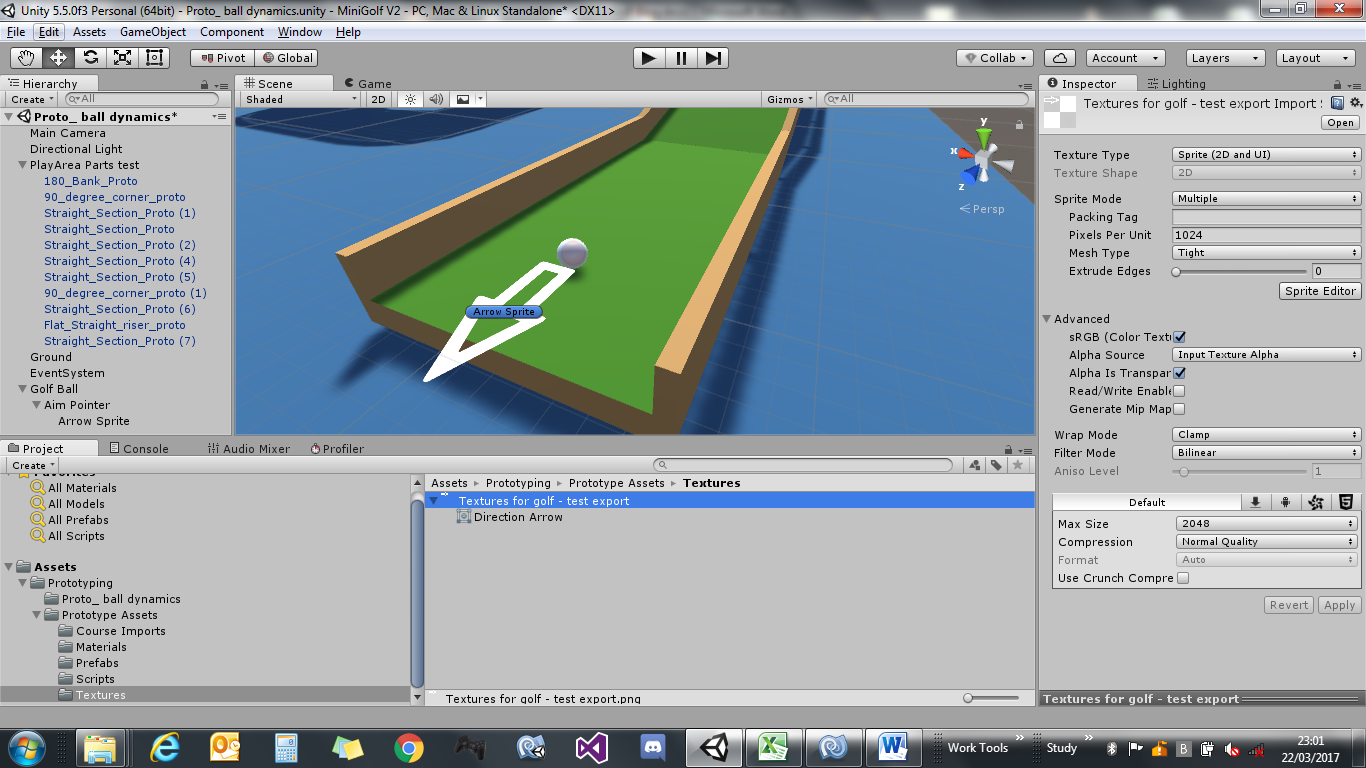
**22/03/17 (1 Hour Spent)**

So my next piece of the puzzle, I think will have to be the ball.

I think today I will have a look at how im going to get the direction to hit the ball in.

Im thinking of just using a simple arrow sprite for a direction indicator. And use raycasting to get the direction. See how that pans out over the day.

Made a little arrow in Paint.NET and exported as a PNG, then imported that into unity as a 2D sprite.



Ive put a new ball (just a unity primitive with a scale of 0.1) in the scene to test size (phew hard part done lol). I think for a sort of fun and not too serious look, this appears about right size wise.

If you look at the hierarchy there for the GolfBall, ive added a child empty ‘AimPointer’ this has the ‘Arrow Sprite’ as a child to it. Just to see how it looks. Not sure how im going to use this ‘Aim Pointer’ yet. If I leave it there, its going to just tumble around when the ball rotates since at this point the balls stationary. See how it goes later.

***Though:*** *well the ball will be stationary when taking a shot, so it might not matter too much, as its not going to be there when the ball starts rolling just need to make sure it ?? and when the ball is stationary it will be visible and always pointing in a direction . hmm*

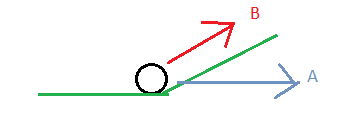
Need to have a think, either leave it there, or try and lock the rotation to just the Z axis, but don’t know how that’s going to work on a child object.. but that’s for another day :)

**29/03/17 (2 Hour Spent)**

The first order of today after housework and a coffee, is getting my mouse input to try and get the firing direction and have the arrow point to where the mouse is at.

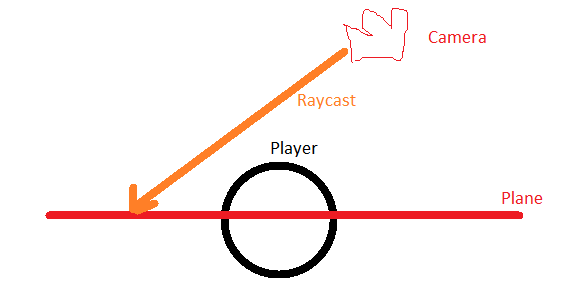
I could try and raycast to the parts of the course, but, there may be times where I don’t want to apply a direction at an angle other than on the horizontal. It would also cause issues if the mouse pointer wasn’t always over a part of the course. So ill remove that out of the equation and just use a ground plane.

The only other thing that I might have to think about later on is if the ball is at the bottom of a slope.

do I apply the force in the B, direction or just stick to A to make things simpler and use the same throughout.

If it causes any issues once I get a prototype done, then I’ll have to look over it again and rethink, but for now, ill stick with the horizontal as the force direction.

Ive done something similar in the past with raycasting where a plane is drawn through the player and I get the hitpoint of the ray, like below.

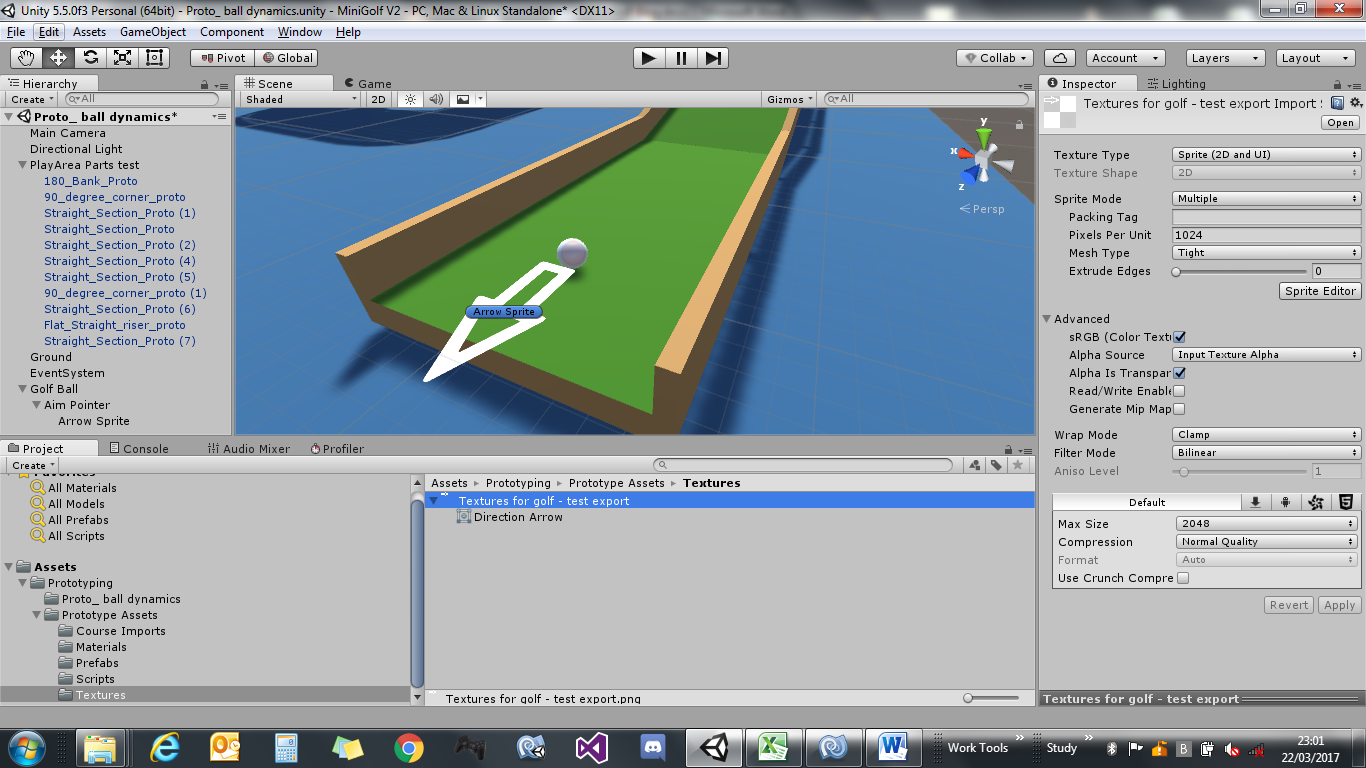


Where I want to find a point on the same Y plane as the ball, and then have whatever im using as a visual pointer look towards it.

Now what I want is this plane to always be straight through the centre of the ball and the plane always horizontal.

Before I go finding out how to rotate anywhere, I need something to rotate.

So heres what im using, and ill explain why and what I was thinking about.



So **GolfBall** is the main ball and that’s going to do all the rolling about etc.

Ive added a child empty, **AimPointer**, this is the one that im going to rotate around, so it doesn’t affect how the ball behaves. Not much good in having an empty with nowt in it.

So ive made a little arrow in Paint, and imported and set to a sprite, and added this sprite as a child to AimPointer. Now ive orientated the arrow sprite to point in the same direction as the Z axis of the AimPointer.

**Reason?** Well, once I find the direction with raycasting, I can use Transform.LookAt() to rotate the AimPointer. And using that lookat(), it always points the forward Z axis to the target, hence the reason the arrow sprite is rotated to point down the Z axis of the AimPointer object. And it wont affect how the ball rotates in case of dodgy physics behaviour.

And to answer another question I had, surely as the ball rolls it will affect the rotation of the arrow?

But, no, not really… because A: the ball will not be rolling when the arrow is visible. And B: the arrow will always be facing towards the mouse pointers raycasted point on the plane so it will always have the normal facing upwards, and the Z axis pointing towards the mouse each frame.

The raycasting bit was ok, the problem I had was how to get the plane to be at the same Y height as the ball and what size to make it, and generally how to draw it.

Well looking at the docs for Plane and how to create one programmatically   
<https://docs.unity3d.com/ScriptReference/Plane.html>

Its as clear as mud on first read, with the constructors

Plane(Vector3 inNormal, Vector3 inPoint)

Plane(Vector3 inNormal, float d)

Plane(Vector3 a, Vector3 b, Vector3 c)

I didn’t have a clue what the first lot were the first time, so I went with the the 3 vector points, and created points around the golfball, to what I thought was creating a plane at that size with the corners I specified for the 3 vectors. But testing it it seemed to work no matter how far away I put the mouse.

So I started with this

        Plane groundPlane = new Plane(  
            *// just need to declare 3 corner points to create the plane, as it iterpolates the 4th corner.*  
            *// doing it this way, creates the plane dead centre of the ball, JUST REMEMBER CW winding for normal.*  
            new Vector3(transform.position.x + 1f, transform.position.y, transform.position.z-1f),  
            new Vector3(transform.position.x - 1f, transform.position.y, transform.position.z-1f),  
            new Vector3(transform.position.x + 1f, transform.position.y, transform.position.z+1f));

But after more readingup, I read the part again at the top of the scripting ref, where it said that the plane was infinite size, so the 3 point method I used just basically declared its position and normal direction.

Back to the scripting reference page for another read once I understood it a bit more and ended up realising I could use this instead

Plane groundPlane = new Plane(Vector3.up, transform.position);

Big difference, a lot more readable and a lot more compacted.

But all that messing about and reading took about an hour to suss out.

The rest of the code was fairly straight forward (grabbed a reference to the AimArrowSprite to use for lookat ) so used the following

(excerpt, full code in repo)

*// used for raycasting out, logs hit distance*  
        float rayDistance;  
  
        if(groundPlane.Raycast(ray, out rayDistance))  
        {  
            Vector3 point = ray.GetPoint(rayDistance);  
            *// so i can see the ray*  
            Debug.DrawLine(ray.origin, point,Color.red);  
            *// since the plane is at same Y level as ball, set the aim arrow to look at the hit point.*  
            *// (always uses the transforms local forward vector, this way the arrow is always flat)*  
            transform.LookAt(point);  
        }

Works well, so tomorrows job, get the ball to move, and switch on and off the targeting arrow.