Perception is a mobile game developed in the Unity game engine using C#. Perception is a puzzle game which utilise the idea of having to change how to look at thing to find the solution. This project was created as a part of one of my University assignments. The assignment was to create a game prototype for any platform of your choice.

The objective of Perception is to build a shape using blocks on a 5x5x5 grid so when all the walls on the outside move inwards the shape pass thought them. /GIF/. This is the core concept of the game as it needs the player to not think in one dimension.

//Challenges

One of the challenges that I encountered was how to create each level. There were a few ways I could have gone with this. The first and most obvious being to create all the wall elements at compile time. This is great for prototyping as I could quickly build new levels. However, there was two reasons I didn’t like the idea of hand placing all the object within the scene. The first being the large number of objects which would be present within the hierarchy and the second being scalability. I wanted an easy way to add new levels without needing to change anything within the editor hierarchy or gameplay systems in fear of introducing new bugs.

The solution which I went with was to have .json files act as levels. Each .json file would contain all the data need for each level, then when a level was selected from the main menu that level .json file would be selected then parsed into a LevelData class which could be used by the LevelManager to generate the level selected.

{% highlight javascript %}

{

"title": "Tap to place a cube",

"wallSpeed": 2,

"wallLifeSpan": 4,

"recomendedCubeCount": 1,

"goals":

[

[ "Time", 20 ],

[ "Score", 2 ],

[ "Moves", 1 ]

],

"wallMap": [

[ "X", "X", "X", "X", "X" ],

[ "X", "X", "X", "X", "X" ],

[ "X", "X", "X", "X", "X" ],

[ "X", "X", "X", "X", "X" ],

[ "X", "X", " ", "X", "X" ],

[ "X", "X", "X", "X", "X" ],

[ "X", "X", "X", "X", "X" ],

[ "X", "X", "X", "X", "X" ],

[ "X", "X", "X", "X", "X" ],

[ "X", "X", " ", "X", "X" ]

]

}

{% endhighlight %}

<pre class="brush: c#">

/// <summary>

/// Level Data class to store all data for a level

/// </summary>

[System.Serializable]

public class LevelData

{

public string Title;

public float WallSpeed;

public float WallLifeSpan;

public int RecomendedCubeCount;

public LevelGoal[] Goals;

public string[][] wallMap;

}

With this setup I can quickly create new levels by adding a new .json file to the levels folder. This also would allow other developers who might not understand the systems in place within the game to add new levels which is important for larger teams.

</pre>

<script type="text/javascript">

SyntaxHighlighter.all()

</script>

//Style

The style I wanted for Perception was minimalistic. This was because puzzle games to me shouldn’t have anything that takes away from the puzzles on screen. I didn’t want players attention wondering from the main mechanic; the puzzle. Originally the wall tiles were going to be 2D images however after some playtesting and reflection having the tiles be 3D adds depth and to me makes the game feel more complete.

Another style consideration was the colours. I again wanted colours that didn’t distract the player but were easy to look at and gave a calm atmosphere.

[Image]

//Tech detail

//Future / Improvements

Looking back on this project I can see some improvements which could be made. Two main ones being, having the blocks be affected by external inputs like physics and having and not having to build the .APK when a new level is added.

In the coming weeks/months I will continue to develop Perception.

When I first completed this project, I had always wanted sound in it. At the time, I did not have any knowledge of what to do. Didn't even know of any sound engines that I could implement within C++. That’s when my lecture told me of FMOD. FMOD is used industry wide and has been used in multiple projects such as BioShock2, Crysis, Hitman Absolution and many more a list can be found [here](https://en.wikipedia.org/wiki/FMOD "FMOD").

FMOD can also be incorporated in to a C++ project. Over the last few weeks I have been researching and have implemented FMOD into my Space Invaders game. Short sound clips are played when you fire a bullet, get hit by a bullet and more. I hope to not only increase the sound library within my Space Invaders but also use the skills and knowledge I have gain from this to incorporate FMOD into more of my C++ projects.

FMOD can be foud [Here](http://www.fmod.com/ "FMOD")