Journal

# 15/10/12

I attended a planning poker session with my project supervisor and the other students who are using Kinect in their project. This helped a lot with identifying oversights in my user scenarios and in narrowing down time estimates for the user scenarios.

# 18/10/12 & 19/10/12

I set up all of my stories and sub-tasks on Jira. I could not set up a sprint as I do not have administration privileges. I have emailed my supervisor in relation to this but will work around it, for now. I downloaded and installed all the required software for development with the Kinect and Xbox360.

# 20/10/12

I went about setting up my Microsoft account for developing for the Xbox360. I could not register my account due to a glitch in the system. I submitted a support ticket containing all the required details and the reproducible steps I took. I shall develop for Windows until this issue is resolved as I cannot deploy Xbox360 projects without this registration. This has and will cause severe delays, due to the loss of time today and the extra time it will take to port the game to the Xbox360 once this is resolved.



# 21/10/12

I created a Windows game to work on while I’m waiting on a response from the Xbox360 developer support team. I implemented the Farseer physics engine to update on-screen sprites, using keyboard input to apply forces. The engine turned out to be easier to implement than I expected. It contains far more helper functions and classes than the average Box2D port. This has helped recuperate some of the time lost yesterday.



I created the input manager to place a layer of abstraction between the game and the input device. This will allow me to debug while the Kinect is not present/available. This will also make the game easier to port to different platforms, should the opportunity arise. I have skeleton code in place for a number of input types but have only fully implemented keyboard input so far.

I researched what was contained in the Kinect SDK API and what functionality I would require from this. I read up on these features to get to grips with what I would be using. I also added Kinect device support to the project. This does not perform input yet but the device is supported and managed correctly.

## 23/10/12

Read up more on implementing polling to retrieve data from the Kinect sensor as this is better suited to games than event driven frame requests. I implemented this on the skeleton stream to track joint positions for any active skeleton in the field of view of the camera. The resources I used for reference here were the “Developer Toolkit Browser” and multiple articles from the msdn Channel 9 Coding4Fun site.

Some of the potential issues which I have identified from this are inaccurate readings for the positions of joints which are not fully visible and trying to identify which skeleton is the correct player skeleton.

## 25/10/2012

Today I started to implement state management logic for menu updating. I also started adding gesture tracking to the project. I used the project at <http://kinecttoolbox.codeplex.com/> as a guide for handling gesture tracking. This project was built for the use with Windows Presentation Foundation so I have had to port the gesture management to suit XNA better. I have only implemented a base gesture detector class so far. I intend to implement a gesture fully over the weekend.

## 26/10/2012

Today I implemented a gesture manager and basic swipe gestures. I used the project at <http://kinecttoolbox.codeplex.com/> as a guide again. I experienced a few minor issues porting the event based code to XNA. I implemented the gesture manager class to deal with the events and allow for simple frame based polling from my input manager.

## 27/10/2012

I changed the structure of the gesture detector and gesture manager again to get rid of the last of the event based logic. I edited my sprite class so each object now handles its own loading. I separated the file paths for images from the main game class into a static class of constants to isolate the hard typed strings from the main game logic.

I implemented displaying of the user's hand active position on screen and the querying of this position for menu selection.

I ran into a number of issues implementing this:

1. I had to separate the displayed hand position from that of the skeleton being tracked so it could be scaled across the screen, otherwise the user could only reach a small portion of the screen.
   1. The original skeletal hand position is then only used to trigger the gestures for selection.
   2. The separated screen hand position is then used when querying what position the user selected.
2. The position of the users hand needed to be offset by the shoulder position to centre it on the screen.
3. The position of the screen hand then needed to be scaled dynamically so that the user could reach all of the edges of the screen.



I researched speech recognition using the Microsoft Speech Platform SDK and implemented basic voice commands in the project. I used “The Purple Book” (<http://channel9.msdn.com/coding4fun/kinect/The-Purple-Book-Using-Kinect-for-Windows-with-XNA>) and the “SpeechBasics-WPF” tutorial, from the Kinect developer toolkit browser, as points of reference when implementing this. I have only added basic voice control so far so have not come across any major problems.

## 28/10/2012

I implemented most of the required menu items and basic menu logic. This is taking a lot longer than I originally estimated. This is mostly due to the additional time required to ensure that both voice and touch input are catered for.