Balloons Project – Client System

Andrew Dodd

# Introduction

This document describes the Balloons Client system in detail and gives reasoning behind some of the design decisions that were made. The Client system is the system which the user interacts with on the screen; it makes use of the Microsoft Kinect hardware to provide natural user input and connects to the Balloons Server to manage the items on screen.

This document only describes the technical aspects of the Client system; for design decisions and specification relating to the user interface and input controllers please consult the Client Design document.

# System Design

As much as is possible, the Client system is split into four components: Network, Graphics, Physics and Input; however these four components also need to be tied up in a central location – XNA puts the burden of this into the Game class. With the exception of Graphics which is part of the Game class, each sub-system has its own code package and is quite discrete from all the other parts, i.e. these components are loosely coupled.

## Network Subsystem

The Network Subsystem implements the INetworkManager interface and is responsible for managing communication between the Client and Server. The default implementation relies on the ScreenConnection class from the Messaging library, details of which can be found elsewhere.

|  |  |  |
| --- | --- | --- |
| **INetworkManager** | | |
| Connect |  | Connects the Network Manager to the Server. |
| MoveBalloonOffscreen | Balloon  Direction  Exit Position  Velocity | Sends a message to the Server telling it the given balloon has moved off screen. The direction is the side the balloon exited, the position a number between 0 and 1 of the height of the balloon, and velocity the velocity of the balloon as it left. |
| NotifyBalloonPopped | Balloon | Sends a message to the Server telling it the given balloon has been popped by the user. |
| GetBalloonDetails | Balloon ID | Retrieves the details of the given balloon from its ID. |
| UpdateBalloonDetails | Balloon | Updates the balloon’s details on the server – these are mainly for decorative purposes as the Client cannot change the balloon’s content. |
| ProcessMessages | On New Balloon  On Pop Balloon  On Content Update  On State Update | Processes messages from the Server and will be called once per Update loop. The four parameters are Actions which are called when the relevant message is received from the Server. |

## Physics Subsystem

[todo]

## Input Subsystem

The Input Subsystem tracks user input and is defined by the IInputManager interface. Two input subsystems are implemented in the project: one for using the Kinect, and one for using the Mouse. The Kinect system is described elsewhere in this report; the Mouse system is very simple in comparison.

When no mouse buttons are pressed, the mouse emulates a single hand which is centred on the position of the mouse. If the left mouse button is held down, a second hand appears with its position relative to the first hand. If, whilst still the left mouse button is still down, the right mouse button is also depressed, then the two hands will emulate a clapping motion.

|  |  |  |
| --- | --- | --- |
| **IInputManager** | | |
| Initialize | Screen Size | Initialises the Input Manager. Some Input Devices use normalised dimensions so the screen size parameter should be stored as the values returned by this Interface should be in screen co-ordinates. |
| GetHandPositions | Returns Hand[] | Returns an array of Hand objects, one Hand per physical hand. A Hand object describes the position of a single hand and has an identifier to tell which user the hand belongs to.  The Hand objects must be static between calls; i.e. a Hand object should represent a physical hand for the duration of said hand being focused as an input entity. If this is not the case, and a new Hand object is generated every frame, the physics engine will not work correctly. |

## Graphics Subsystem

[todo]

## Game System

[todo]

# Development Report

Throughout the development cycle of the project, several builds were presented to the client. In the first few weeks, many of the demo builds were individual feasibility studies of individual components needed to build the final project. This section gives details of what requirements were implemented in the Client system for each week.

## Client Demo 1 – 7th February

The first prototype presented to the client showed hand tracking using the Kinect and the capabilities of the physics engine using mock graphics. The client thought that the Kinect input seemed to work well and was generally pleased with the input at this stage.

Another prototype build during this week’s meeting demonstrated the design of the user interface which the user was also pleased with; however that prototype was discarded as it was easier to implement the design on top of the input prototype.

## Client Demo 2- 14th February

The second demo prototype was much more advanced than the first. Many of the UI concepts were implemented including balloons along with buckets to provide the “painting” functionality. The mock graphics used in the first prototype were replaced by nicer images. The balloons were already tied into the physics however the rules were tweaked to make them much more realistic. The ability to view the content of a balloon was also added by putting in a “pop balloon” action in the Input interface, which when called, caused a content box to open with the desired content. Balloons were also given a small “tag” with a short text description.

The Kinect input system was now governed by the physics engine. Whilst this allowed for collision detection, the user input lag added caused the client some distress as it ran very slowly on the target hardware. Furthermore, the client had issues reaching the top of the screen on the target hardware.

Finally, the networking features of the system were added in this build. These allowed balloons to be generated by the Server and pushed down to the Client system. Balloons could be pushed between screens although some bugs in the server made this difficult to demonstrate.

## Client Demo 3 – 21st February

Several bug fixes and improvements were implemented for this build. Crashes from the previous build were ironed out and fixed; the issue with the user being unable to reach the top of the screen was also fixed. Text-wrapping on the balloon tags was fixed and the tag boxes made to fit the text once it had been wrapped to remove any extra white space.

The content box was improved and made to show images and a QR Code which encoded the URL of the content article, allowing a user to visit that link with a mobile device. A timer was added to the content box showing the user how long until it automatically closed and a large button was placed in the top right of the screen which the user could use to close the box themselves. An HTML version of the content box was also implemented, it was planned to replace the hard coded versions in the final release.

Overall the client was very pleased with the improvements made to the system. Unfortunately it seemed impossible that the Client System could be run on the target hardware; this demo was shown using a laptop hooked up to the screens rather than the hardware.

## User Evaluation Build – 27th February

During the user evaluation some fixes were applied to prevent concurrency issues in the HTML renderer when rendering images; other than this the Client system’s build was unchanged from the previous week.

## Week 9

New development cycle?

# Third-Party Components

The Client system makes use of several third-party components to reduce the amount of development required:

* *Microsoft XNA Game Studio 4.0:* An environment for quickly creating games which provides several library functions such as loading content and drawing to the screen.
* *Microsoft Kinect SDK 1.0*: The official SDK providing the ability to interact with the Kinect hardware.
* *Farseer Physics 3.3*: A physics library derived from Box2D, a very popular physics library on which many iPhone and Android games are based. It is used to provide collision detection and object interaction between balloons and gives a realistic feel to balloon movement.
* *Terra Informatica HTMLayout*: A lightweight HTML renderer used to render the labels and content boxes of balloons. This allowed us to provide much richer content than the built-in text rendering abilities of XNA.
* *ThoughtWorks QRCode*: A QRCode generation library, used to convert content URLs into a format which can be quickly recognised by a user’s device.