Multi-User Drawing Surface

MSU Computer Science and Engineering Capstone Team Members:

Robert Meyer

Kirsten Partyka

Sean Murphy

Charles Otto

Microsoft Team Members:

Shailesh Saini

Maciej Skierkowski

Date: April 4, 2008

1. Executive Summary

Multi-User Drawing Surface (MUD Surface) allows users to collaborate online using a virtual whiteboard. Like a real whiteboard, users can share ideas by drawing shapes, lines and adding text, but because it's online, users can work together regardless of their physical locations.

The primary goal of the MUD Surface is to give people a way to collaborate like they would using a whiteboard with each other in person, when it is not possible for them to physically be together. The process is designed to be as simple as it can while still providing users with the tools they need to perform useful collaboration in real time. Some of the tools which will be discussed in detail later include tools for creating shapes, adding text either by itself or to an existing shape and the tools necessary to edit existing shapes and text.

Example Scenario

Joe, Sally, Ben and Lisa are working with each other on a project; however, they are unable to meet in person. They want to begin building a model for how they will design a new software application, but creating diagrams and exchanging them through email is too tedious. In addition to the problems associated with using email to exchange documents, only a single person would be able to change the diagram at a time. If multiple people were editing the same revision of a document simultaneously, conflicts would arise when trying to merge all the changes together later.

To solve this dilemma, the group decides that it is best to use MUD Surface. Joe is the first to visit the website. Joe notices that the 'users' area shows only one user, himself. Soon thereafter Sally, Ben and Lisa visit the website. Everybody's screen is updated to show all four group members.

Lisa begins drawing the overall architecture of the software application which has three main areas: User Interface, Client Backend and Data Store. Because Joe has a lot of experience with Data Stores, he begins adding necessary components to the diagram. Simultaneously, Sally and Ben begin flushing out other parts of the diagram. Ben notices that Sally made a typo and corrects it for her. Just after correcting it, everyone receives the correction.

The group continues diagramming for another 30 minutes before they decide to take a break. Everyone closes Internet Explorer knowing that they can simply visit the website later and the diagram will still be there.

2. System Architecture

The system will be comprised of two main components, the server and the client. The client will be accessed by the users through a website available anywhere. This

client will allow the user to interact with the whiteboard and will also handle communication with the server in the background. The server will listen for client activity and notify the connected clients of any changes. This is illustrated in figure 1 below.

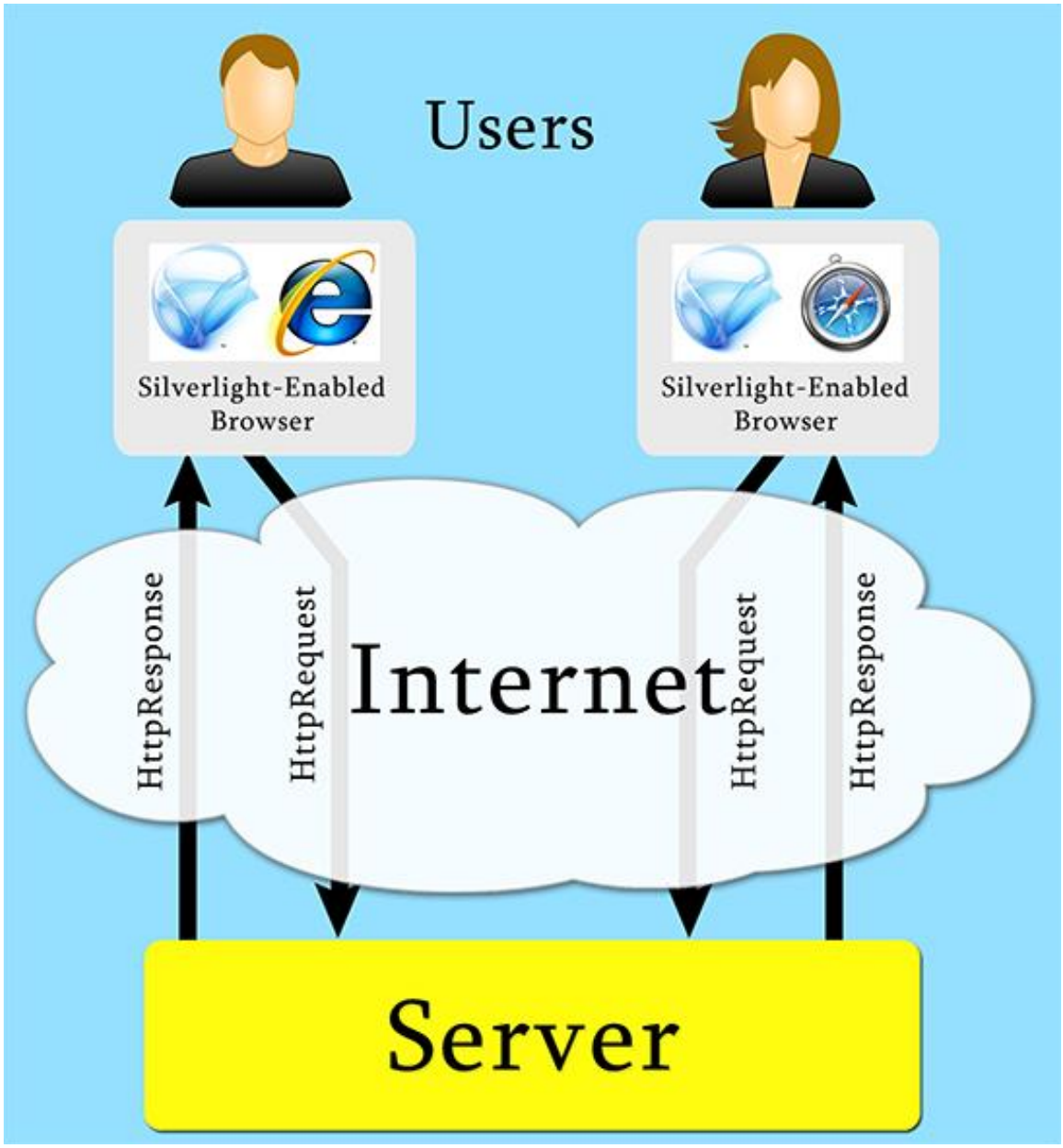


Figure 1: Architecture Diagram

3. Features and Requirements

3.1. Overview

The strategy being used for the development of this project is one that

focuses on implementing a basic set of features that have been deemed high priority, and then later integrating more features as time allows. Features have been marked as High priority, meaning they must be included for the delivery, Medium priority, meaning they should most likely be included in the delivery, and Low priority, meaning they may be included if time allows.

Furthermore, the features have been organized into two groups. First are the features that have been agreed on by the developer and the clients, and second are features that may be present in the project at some point.

3.2. Features

3.2.1. Shape Drawing and Editing

Priority: High

The user will be able to use buttons on the toolbar to select a shape to draw on the canvas. The shape will be drawn using mouse clicks and movement as input. Once the shape has been drawn on the canvas, the user will be able to select it and edit its position, properties, and appearance.

3.2.2. Text Addition and Editing

Priority: High

The user will be able to add text to the canvas. Once the text has been added, the user will be able to select the text and change its position, properties, and contents.

3.2.3. Deletion of Objects

Priority: High

The user will be able to delete any object they are able to select.

3.2.4. Canvas Updating

Priority: High

The user will be able to see any changes being made by other users in as close to real time as possible. Any objects that are currently being changed will be highlighted in the editor's color, and shown with a padlock to show other users that they cannot access that object at that time.

3.2.5. Freehand Drawing

Priority: High

The user will be able to use a tool to draw freehand lines and shapes using the mouse. These lines will be treated as objects once drawn and will be editable as such.

3.2.6. Multiple User Interaction

Priority: High

The Multi-User Drawing Surface application can be used by many users

simultaneously. To provide the best user experience possible, several measures have been taken to assist users when working with others. One such measure is that users are assigned defined colors (as displayed in the sidebar). When a particular user is editing an item, the item will be highlighted with the color of the user as a visual cue to indicate to others that they should not also attempt to edit the same item. The item will be locked to other users until the edit is finished – they will be unable to select the item. The expected number of concurrent users is four to five.

3.2.7. Conflict Resolution

Priority: High

In cases where network latency is higher than normal, a conflict could occur in which two users try to edit the same item at the same time. The first request to lock the item that is received will be honored by the server, and will send a message to all clients to lock it. If another client receives a message to lock an item that they are currently attempting to edit, they will receive a notification that another user is already editing the item, and the item will be returned to its original state, reverting their changes. The locked visual cue will then be displayed on the item. The user must wait until the lock has been released (when the other user unselects the item) to make changes.

3.2.8. Colors

Priority: Medium

The user will be able to change/set the color of all items.

3.2.9. Text in Shapes

Priority: Medium

Any shape that is not already a text object will be able to have text associated with it. This text will appear within the shapes. This text will be editable just as any other text box but will move and be deleted along with the shape it is associated with.

3.2.10. Login

Priority: Low

Users will enter a name to identify themselves with in a popup window when the program is first launched.

3.2.11. Notification System

Priority: Low

When it is helpful to notify the user of something, the application may notify the user using the notification system. Unlike intrusive message boxes, the notification system is to be designed in such a way that the user is not interrupted by the message; however, the message is easily noticed and read.

3.2.12. Layers

Priority: Low

Items may be superimposed on one another. When a new item is created, it is placed on the top-most layer. The user may change the layer of an item by using the user interface. The four layer-change options are send to front, send to back, move forward one layer, and move backwards one layer.

3.2.13. Resource Download

Priority: Low

Since the application uses a number of image files for user interface graphics, there is a period of time when resources are unavailable to the application before downloading is complete. To minimize the amount of time that the user has to spend waiting, an asynchronous download should begin as soon as the user is show the login screen. While the user is entering a username, the download can continue in the background. A progress bar should also be shown with estimated time until completion.

4. Client

4.1. Overview

Users will interact with the client application in their browser window. Multiple users can be connected to the same virtual whiteboard at a time. Users can create and modify their own VisualItems or those of the other users at the same whiteboard.

In Figure 2 below, a use case diagram depicts the usage of the Multi-User Drawing Surface application.

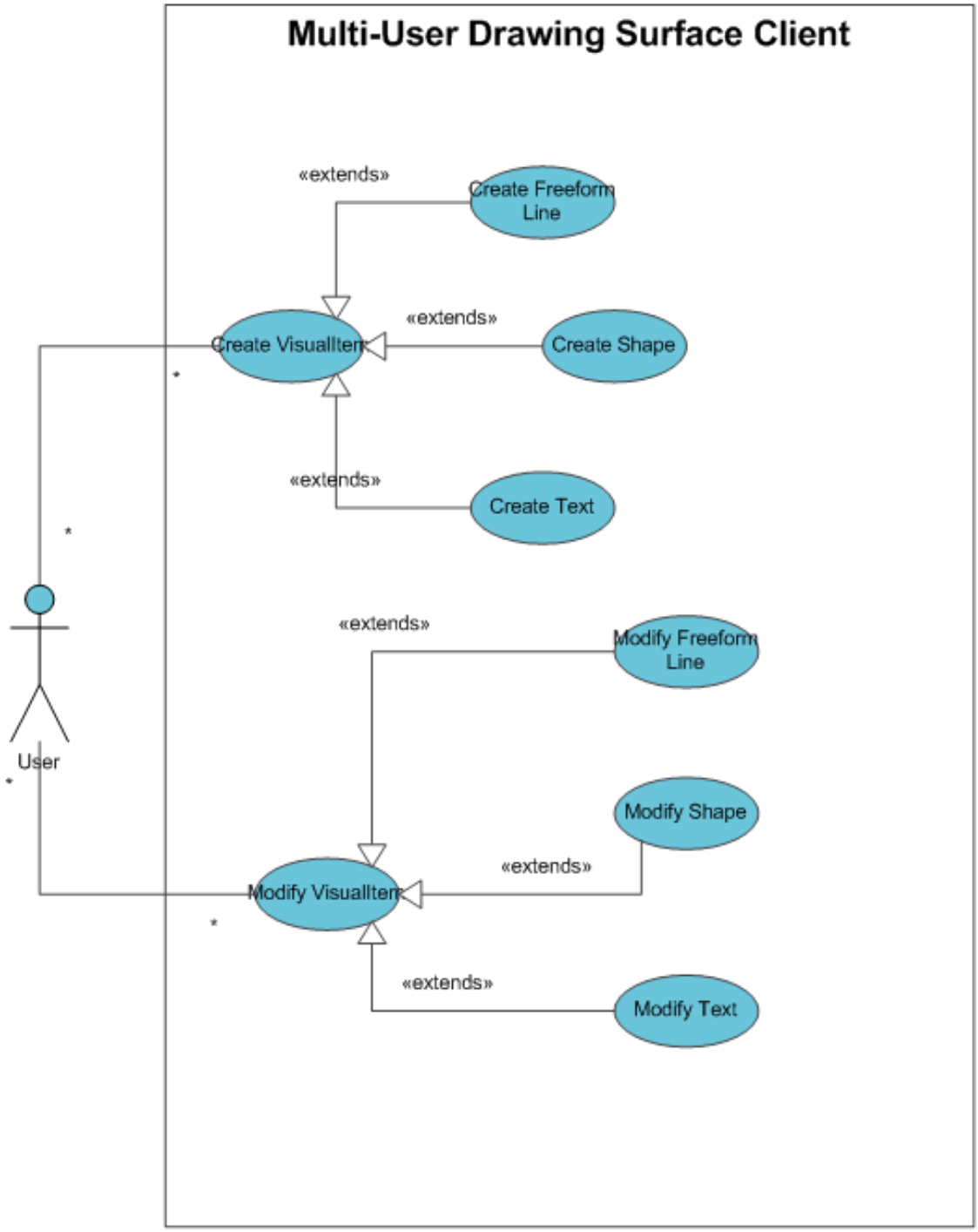


Figure 2: Use Case Diagram

As you can see in Figure 2, users can create VisualItems (shapes, freeform line, text), and they can modify VisualItems.

4.2. Graphical User Interface

The Graphical User Interface is designed with a quality user experience in mind. The

interface is intuitive and simple. A mockup of the user interface with outlined control components is shown in Figure 3.

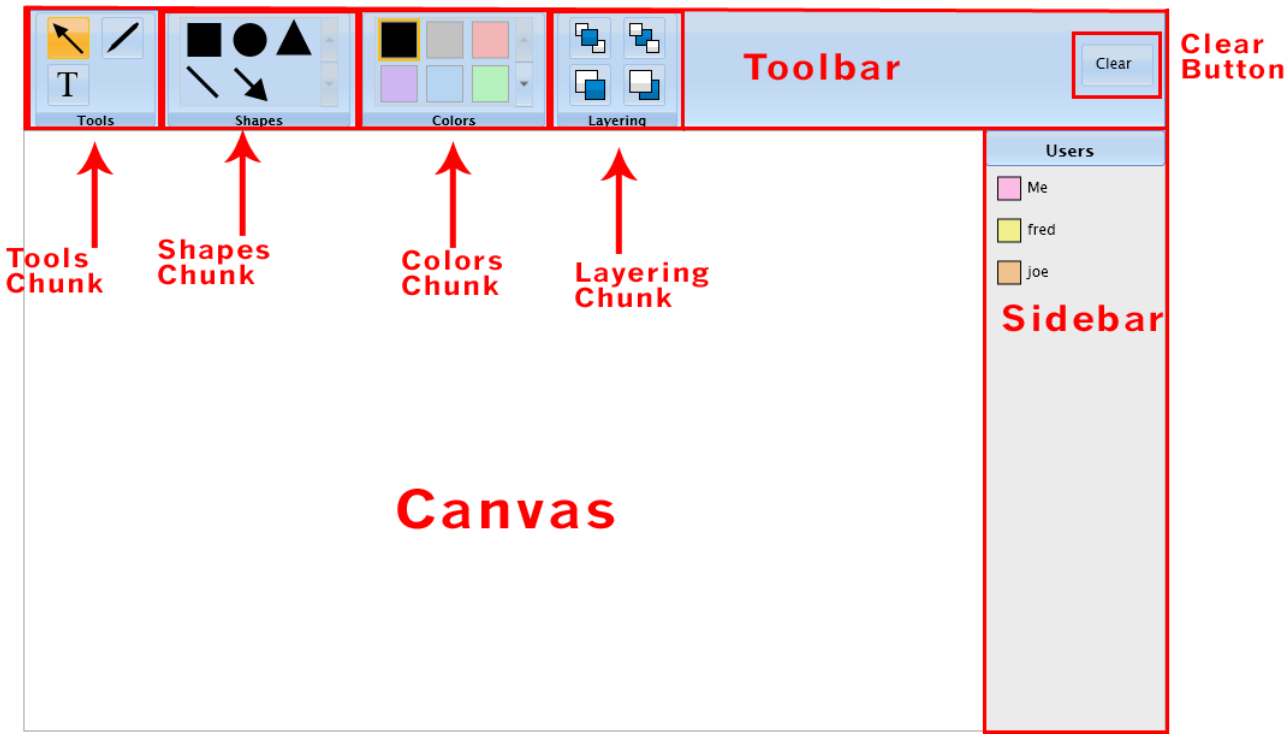


Figure 3: Mockup with the major UI components outlined

4.2.1. Toolbar

The **Toolbar** contains buttons for the key features of Multi-User Drawing Surface. It is divided up into several panels, called **Chunks** for better organization. Room is left for additional Chunks and tools to be added as the application feature-set is expanded.

4.2.1.1. Select Button

The Select Tool is enabled by clicking on the Select Tool Button which is represented by a mouse pointer icon.

Select Tool Button

When this tool is selected, a user can manipulate VisualItems on the canvas. Clicking on the desired VisualItem, and moving the mouse to the center of an item until the "move" cursor is displayed. Finally, the user may click and drag the mouse to move VisualItems.

Move cursor

Shapes and text boxes can be resized. The user must first click on the desired VisualItem, which will highlight the item and allow resize handles to appear. The user can then hover the mouse over a handle until the “resize” cursor is displayed, then drag in the desired resize direction.

Resize cursors

While a VisualItem is selected, the user may press the delete or backspace key to remove the VisualItem from the canvas.

4.2.1.2. Text Tool

The Text tool is enabled by clicking on the Text Tool Button which is represented by a serif T icon.

Text Tool Button

When this tool is selected, a user can add text to the canvas by clicking on an empty space in the canvas. A text box will appear with a caret in it, and the user can immediately begin typing. The user can also edit existing text on the canvas by hovering over a pre-existing text box or text-area of a shape and clicking inside. A blinking caret should appear in the text box and the user can begin typing or editing.

4.2.1.3. Freehand Tool

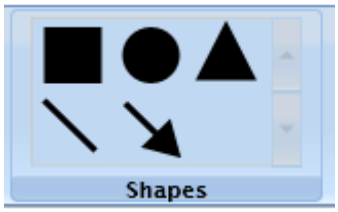
The Freehand tool is enabled by clicking on the Freehand Tool Button which is represented by an icon of a pen.

Freehand Tool Button

When this tool is selected, a user can draw a freeform line on the canvas by clicking and dragging the mouse around the canvas. When the user releases the mouse the line will be finalized.

4.2.1.4. Shapes Chunk

The Shapes Chunk consists of a library of all available shapes. The shapes are located in a scrolling list box.



Shapes Chunk

To draw a shape, the user must click on the desired shape to select it. A user can then draw the shape on the canvas by clicking (this point will form the one of the shape's corners) and then dragging. When the mouse is released, the shape's size will be finalized.

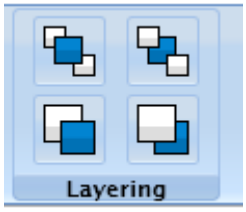
4.2.1.5. Colors Chunk

The Color Chunk displays all available colors in a scrolling list.

Colors Chunk

The current color is highlighted and outlined. All VisualItems drawn on the canvas will use the color that is currently selected. If a user clicks on a color while a VisualItem is selected, the color of the selected VisualItem is updated to reflect the color change.

4.2.1.6. Layers Chunk



Layers Chunk

The Layers chunk allows the user to alter the layer a selected VisualItem. There are four buttons which represent the options for changing the layer of a selected Visualtem.

Send To Back

This allows the user to send the selected VisualItem to the very back-most layer. The VisualItem will appear to be behind all others.

Send To Front

This allows the user to send the selected VisualItem to the very front-most layer. The VisualItem will appear to be in front of all others.

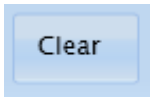
Move Backward

This allows the user to move the selected VisualItem back one layer.

Move Forward

This allows the user to move the selected VisualItem back one layer.

4.2.1.7. Clear Button



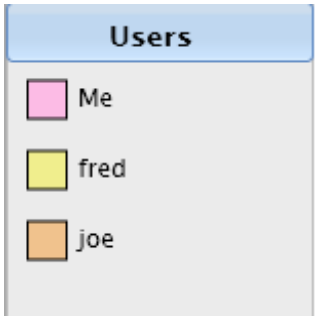
The clear button allows a user to wipe the Canvas clean of all shapes, except for those that are locked by other users.

4.2.2. Canvas

The Canvas is where all VisualItems can be drawn and manipulated. It has a fixed size to allow all users to see the same canvas regardless of screen resolution.

4.2.3. User Presence Pane

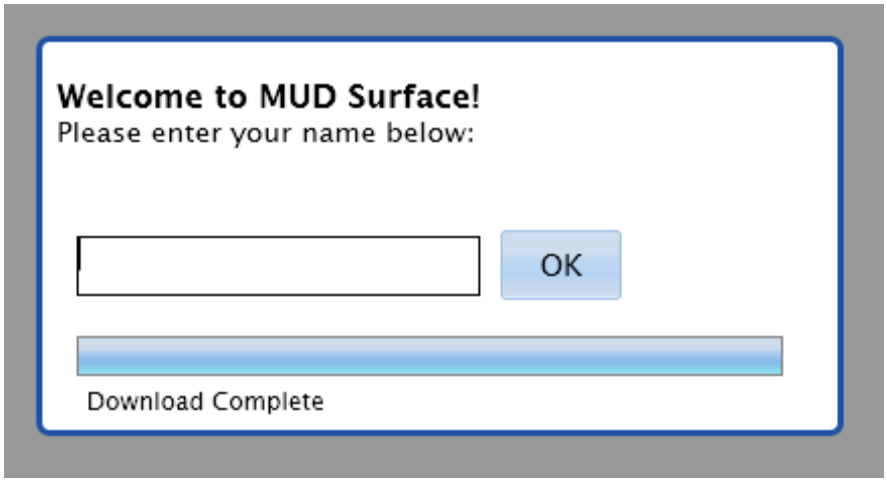
This pane is located in the sidebar. It displays information about which users are connected to the session. This information will always appear in the pane, and will be updated in real-time. Users are listed, with a color next to their name – this color is assigned to a user when they join a Multi-User Drawing Surface session. More about this color identification appears in section 3.2.6 Multiple User Interaction.



User Presence Pane

4.2.3.1. User Login

The User Login is displayed when a user first opens the Multi-User Drawing Surface. The user must enter some form of identifying name into the box, and press the OK button to continue. A progress bar is displayed to show the progress of downloading resources related to the application.



User Login

4.2.3.2. Multiple User UI Elements

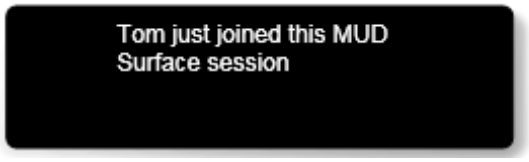
As described in 3.2.6 Multiple User Interaction, many users can be logged on concurrently and working on the same VisualItems. To assist with this there are some UI elements provided.

Locked Graphics

"Locked" Graphic

As previously mentioned, VisualItems that are being edited by a user will be shown to be "locked" to all other users. The example above shows a rectangle with this "locked" graphic.

4.2.3.3. Message Display



Message Display Notification

The Message Display allows the notification system to provide textual information to the user in an unobtrusive way. The example above shows a message that would appear when a user named Tom joins.

4.3. Implementation

4.3.1. User Interface

The Client will be implemented using the design in the class diagrams in Figure 4.

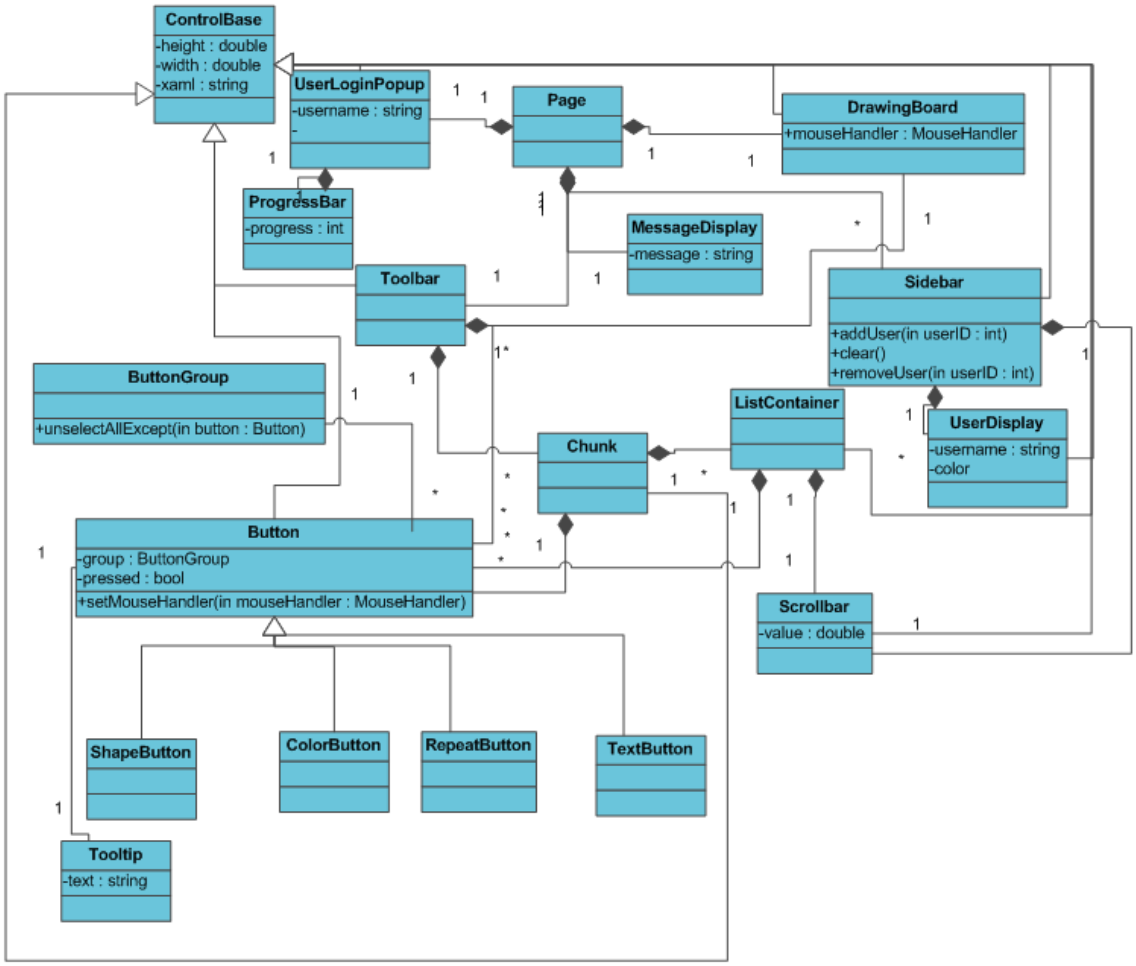


Figure 4: Class Diagram of UI Elements

The UI elements are shown in the class diagram in Figure 4. Below they will be explained in more detail.

4.3.1.1. Page

Page is the main class that all UI components are added to.

4.3.1.2. DrawingBoard

DrawingBoard is the Canvas that all VisualItems are added to.

4.3.1.3. Toolbar

The Toolbar is a UI element that contains the Chunk controls, and is responsible for the layout of the chunks. Toolbar inherits from ControlBase.

4.3.1.4. Sidebar

The Sidebar is the UI element that is the User Presence Pane, and has a list of UserDisplay elements. Sidebar inherits from ControlBase.

4.3.1.5. MessageDisplay

MessageDisplay is a UI element that displays messages on the screen in a transparent black box. This is used with the notification system. MessageDisplay inherits from ControlBase.

4.3.1.6. UserLoginPopup

The UserLoginPopup is an UI element that allows the user to log in via a text box and OK button. UserLoginPopup inherits from ControlBase. The UserLoginPopup box also displays a progress bar for resources that are downloaded asynchronously, so as to indicate to the user when the application is fully ready.

4.3.1.7. ProgressBar

ProgressBar is a reusable UI element that indicates with a bar, the percentage completion of a task. It is used in UserLoginPopup.

4.3.1.8. ControlBase

ControlBase is the main base class for all controls that use XAML to display them. It is responsible for height, width, children elements, and reading in the XAML.

4.3.1.9. Chunk

Chunk is a UI element which goes on the Toolbar and contains a collection of Buttons or a ListContainer. Chunk inherits from ControlBase.

4.3.1.10. Button

Button is a UI element that generates click events when clicked. It has selected, hover, and deselected states. Button inherits from ControlBase.

4.3.1.11. ButtonGroup

ButtonGroup is a class that allows Buttons to maintain mutually exclusive relationships – of all Buttons in a ButtonGroup, only one can be selected at a time.

4.3.1.12. ListContainer

ListContainer is a UI element that contains a list of Buttons. It has a Scrollbar, and has rows and columns of the Buttons inside it. ListContainer inherits from ControlBase.

4.3.1.13. Scrollbar

Scrollbar is a reusable UI element that consists of two RepeatButtons and allows the user to change its value up and down. It is used with ListContainer to scroll up/down a list. It is also used with Sidebar to scroll up/down the list of users. Scrollbar inherits from ControlBase.

4.3.1.14. ColorButton

ColorButton is a Button specifically styled for displaying colors. ColorButton inherits from Button.

4.3.1.15. ShapesButton

ShapesButton is a Button specifically styled and customized for displaying shapes. ShapesButton inherits from Button.

4.3.1.16. TextButton

TextButton is a Button that displays text on it. TextButton inherits from Button.

4.3.1.17. RepeatButton

RepeatButton is a Button that is styled with up/down arrows for the Scrollbar. It pulses its 'click' event when it is held down. RepeatButton inherits from Button.

4.3.1.18. Tooltip

Tooltip is a UI element to identify the meaning of icons more clearly to users when they hover over a button.

4.3.1.19. UserDisplay

UserDisplay is a UI element that contains a username and color that is associated with them. A list of UserDisplay items are placed on the Sidebar. UserDisplay inherits from ControlBase.

4.3.2. User Interaction

The following diagram shows all classes related to creation and modification of VisualItems.

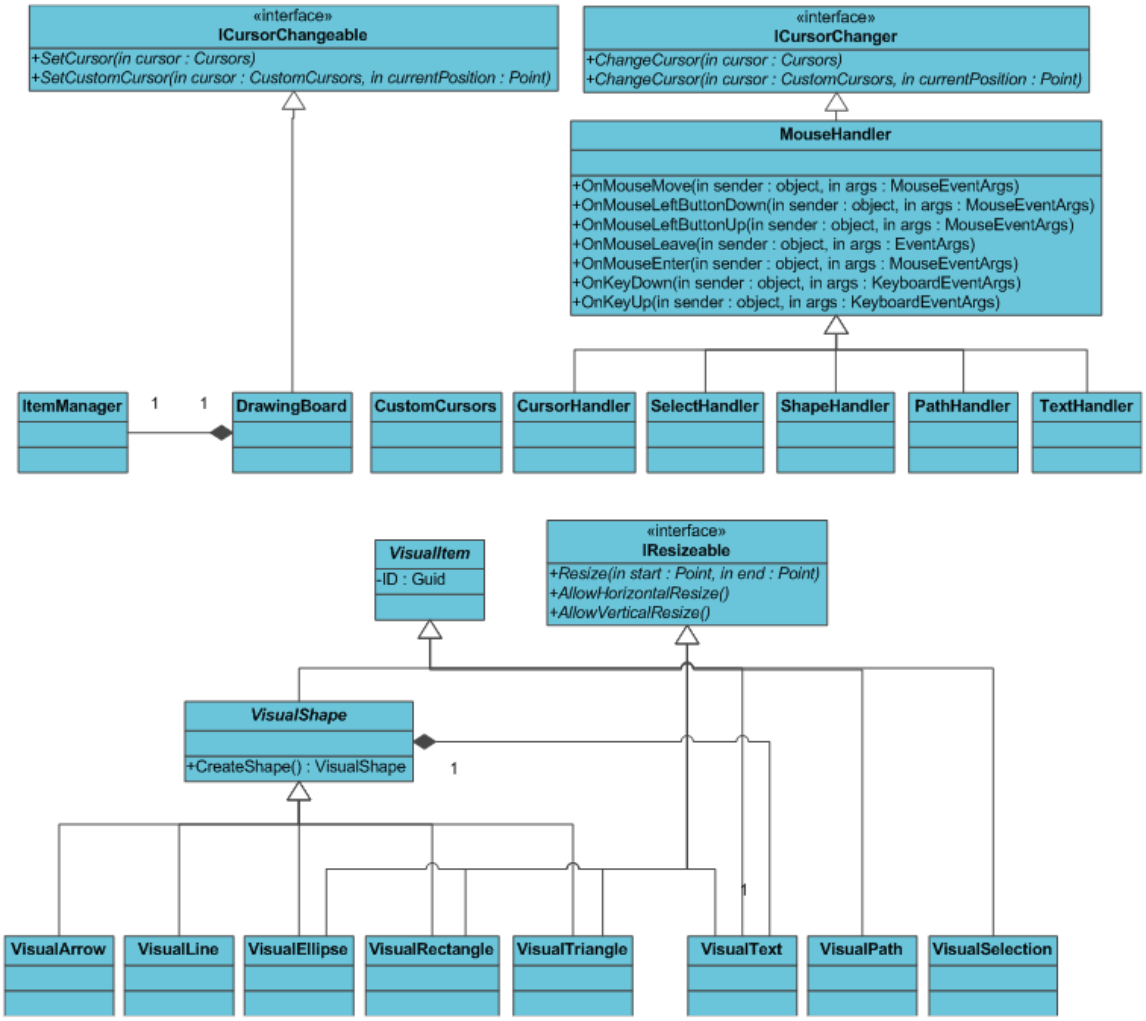


Figure 5: Class Diagram of Classes Supporting User Interaction

4.3.2.1. CursorHandler

CursorHandler is a class designed to change the appearance of the cursor. CursorHandler inherits from MouseHandler so that it may receive mouse events and stay in sync with the application. This class is a result of the inability to specify custom cursor graphics in Silverlight 1.1 alpha and should be deprecated should this application ever be ported over to a newer version of Silverlight.

4.3.2.2. ICursorChangeable

ICursorChangeable is an interface that may be implemented by any control that wants to use custom cursors.

4.3.2.3. ICursorChanger

ICursorChanger is an interface that may be implemented by any class that needs to be able to change the cursor that is displayed inside a control that implements the ICursorChangeable interface.

4.3.2.4. IResizeable

IResizeable is an interface that provides necessary methods to resize a VisualItem. Classes that implement this interface may define the ways in which they may be resized (vertical, horizontal or both.)

4.3.2.5. ItemManager

ItemManager is a class which handles all the logic for adding VisualItems to the DrawingBoard, selecting VisualItems,

4.3.2.6. MouseHandler

MouseHandler is the base class for handling mouse events on the DrawingBoard. It is abstract.

4.3.2.7. PathHandler

PathHandler handles mouse events for drawing freeform lines. PathHandler inherits from MouseHandler.

4.3.2.8. TextHander

TextHandler handles mouse and keyboard events for creating and editing VisualText items. TextHandler inherits from MouseHandler.

4.3.2.9. SelectHandler

SelectHandler handles mouse events for selecting, resizing, and moving VisualItems. SelectHandler inherits from MouseHandler.

4.3.2.10. ShapeHandler

ShapeHandler handles mouse events for creating VisualShapes. A VisualShape must be able to be created using two points (the start and end points of the user's mouse drag event.) ShapeHandler inherits from MouseHandler.

4.3.2.11. VisualItem

VisualItem is the base class for all visual objects that will interact with the ItemManager. VisualItems can be uniquely identified using a Guid. All classes that inherit from VisualItem must implement a number of methods such as one to determine if a point is contained within the visual, set the color of the visual, and a means to move the visual.

4.3.2.12. VisualShape

VisualShape inherits from VisualItem and remains an abstract class. It provides

generic code that can be used for any VisualItem that may be created using the ShapeHandler. All VisualShape objects also contain a VisualText object.

4.3.2.13. VisualArrow

VisualArrow inherits from VisualShape and looks like an arrow.

4.3.2.14. VisualEllipse

VisualEllipse inherits from VisualShape and looks like an ellipse.

4.3.2.15. VisualLine

VisualLine inherits from VisualShape and looks like a line.

4.3.2.16. VisualRectangle

VisualRectangle inherits from VisualShape and looks like a rectangle.

4.3.2.17. VisualTriangle

VisualTriangle inherits from VisualShape and looks like a triangle.

4.3.2.18. VisualPath

VisualPath inherits from VisualItem and displays a free form line defined as an ordered list of screen coordinates. A line is drawn connecting each of the screen coordinates.

4.3.2.19. VisualSelection

VisualSelection inherits from VisualItem and is the graphic that is shown when a user is selecting multiple items using a drag-select. It is used exclusively by SelectHandler and is not to be a VisualItem that is permanently added to the DrawingBoard.

4.3.2.20. VisualText

VisualText inherits from VisualItem and contains text that may be edited. Instances of VisualText are created by the TextHandler. TextHandler may also be used to select an already created VisualText object so that its text may be manipulated.

4.3.3. Client Backend

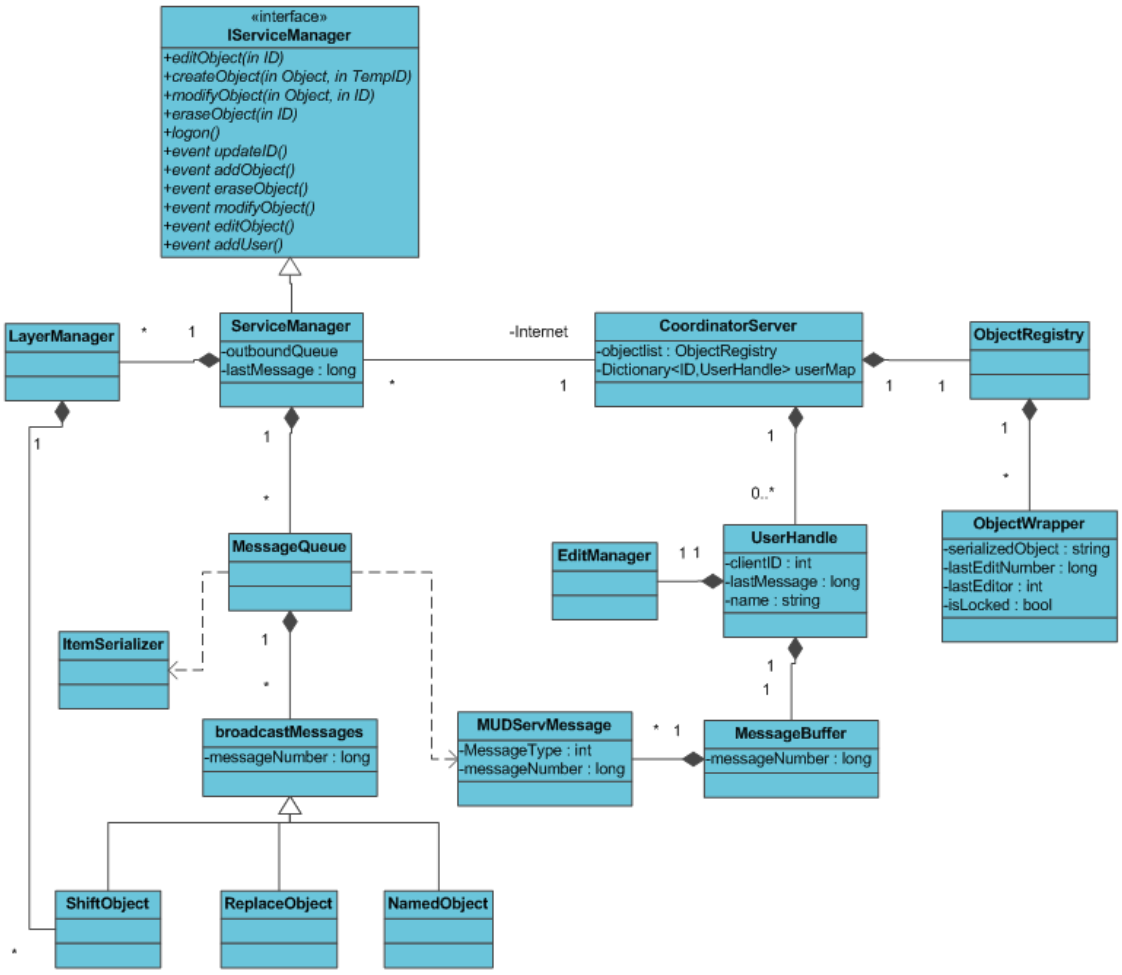


Figure 6: Class Structure of the Server and Related Client Backend

4.3.3.1. RealServiceManage

RealServiceManager implements the IServiceManager class in a manner that allows communication between the client and the server. It handles outgoing messages by using the MessageQueue class, and handles incoming messages by calling events. It also handles layering synchronization concerns using the Layer Manager class.

4.3.3.2. MessageQueue

MessageQueue stores any outgoing messages given to it by the RealServiceManager, and provides a compressed batch of messages to actually be sent to the server.

4.3.3.3. LayerManager

LayerManager keeps track of layer changes that have been sent to the server but not yet acknowledged. It also detects any conflicts caused by layer changes received from the server and handles them by undoing layer changes invalidated by the conflict.

4.3.3.4. ItemServiceManager

ItemServiceManager is an interface which any class that represents server communication must implement. It is implemented by RealServiceManager and a number of other classes that were only used for testing.

4.3.3.5. ItemSerializer

ItemSerializer performs custom serialization and deserialization for all VisualItem classes.

5. Server

5.1. Overview

The Multi-User Drawing Surface server provides synchronization and conflict resolution for multiple clients connecting to a virtual whiteboard. The server relays changes made to the whiteboard by one client to all other connected clients and acts as a conflict resolution mechanism to guarantee that all clients see the same whiteboard state.

5.2. Interaction with Clients

The IServiceManager interface in Figure 6 specifies the client interface to the server. The main client-server communcation mechanism is sending and receiving objects of the MUDServMessage class via the SOAP protocol. This class is a simple wrapper that contains the type of the message and some fields for data contained in the message.

There are enumerated message types for every operation that changes objects on the whiteboard as well as messages types for adding and removing users. When clients make changes to their local whiteboards they send MUDServMessages to the server describing those changes, the server interprets those messages and then relays them to the other clients. When a new client connects to the server it is sent messages describing all objects on the whiteboard as well as messages listing all users currently connected to the whiteboard.

In addition to acting as a relay point between clients, the server also guarantees that clients all have the same whiteboard state. This accomplished by an object level locking mechanism. When a user edits an object it first sends a message to the server requesting exclusive access to that object. If another client has already locked the object the server sends the original client a message indicating that the lock request was denied. If the lock request is granted the server relays that information to the other connected clients.

5.3. Implementation

Due to the tightly coupled class design required when designing a client-server model, the classes described below have been included in Figure 6: Structure of the Server and Related Client Backend in Section 4.3.3 Client Backend.

5.3.1. CoordinatorServer

CoordinatorServer processes and synchronizes messages from clients. It maintains the list of current users and stores data about each user; it also maintains the master list of object on the whiteboard.

5.3.2. ObjectRegistry

ObjectRegistry contains a set of serialized objects that are transmitted to new clients when they log on.

5.3.3. ObjectWrapper

ObjectWrapper stores meta data about serialized objects including their locked/unlocked status, the user who locked them, and the number of the last message to affect the object.

5.3.4. UserHandle

UserHandle stores data about a specific user including the users' numerical ID and name. This class also contains the set of messages that are waiting to be sent to the client.

5.3.5. MessageBuffer

MessageBuffer maintains a queue of messages waiting to be sent to a client. It implements a message compression scheme which checks for and eliminates redundant messages in the queue, thereby improving the performance of client-server communications.

5.3.6. EditManager

EditManager maintains a list of edits that have been sent to a client but which the client has not yet acknowledged. This is used to detect conflicts in some cases.

5.3.7. MUDServMessage

MUDServMessage is a simple class which contains an enumerated message type, and may contain a serialized VisualItem.

6. Technologies

This project consists of three primary components: a client application, a server application and a web server used to send the client application upon request. Each of these components will be developed with a specific set of technologies in mind. C# will be written in Visual Studio 2008 for both the client and server application. The client application will be a Silverlight application built with XAML using the Silverlight Alpha 1.1 Refresh SDK. The server application will be an ASP.NET Web Service Application. The web server will be an ASP.NET website running on Windows Server 2003 and IIS7.

7. Constraints

This application is being built with Microsoft's Silverlight Alpha 1.1. Due to the technology of this application, there are a number of constraints that play a large role in the design of this project.

7.1. Custom Controls Required

Microsoft has not yet implemented control or layout functionality such as buttons, text inputs, or scrollbars. This will all be implemented by hand.

7.2. No Socket Support

There is no socket support in Silverlight Alpha 1.1, therefore all communication with the client and server must be done using HTTPRequests.

7.3. No XML Serialization

XML serialization is missing from the .NET subset provided in Silverlight 1.1 Alpha, JSON serialization must be used.

NOTE: As of March 5, 2008, the Silverlight 1.1 Alpha has been replaced by Silverlight 2.0 Beta. Microsoft has encouraged all future Silverlight applications to be developed with that version of the technology.

8. Schedule

8.1. Overview

2/18 Alpha demonstration
3/17 Beta demonstration
4/21 Project video complete
4/25 Design Day

8.2. Detailed

- Week 1: Jan 20 – Jan 26
  - UI - Kirsten
    - Learn how to build custom controls (Jan 23)
    - Plan UI architecture (Jan 26)
  - User Interaction - Rob
    - Learn about mouse events / keyboard events (Jan 23)
    - Plan interaction behavior (Jan 26)
  - Client Backend – Sean
    - Define server-client communication system (Jan 26)
  - Server – Charles
    - Look into .NET Web Services (Jan 26)
- Week 2: Jan 27 - Feb 2
  - UI – Kirsten

        - Buttons – idle, pressed and hover states (Jan 30)
        - Toolbar and Chunk layout elements (Jan 30)
        - Button groups – maintain mutually exclusive relationships (Feb 2)
        - Create Listbox control with scrollbar (Feb 2)
    - User Interaction – Rob
        - Create shapes classes (Jan 30)
        - Add shapes to DrawingBoard (Jan 30)
        - Learn about free-form and text solutions (Feb 2)
    - Client Backend – Sean
        - Communication grammar defined
            - Object created (Jan 30)
            - Object changed - includes deletion (Jan 30)
            - New user – logon (Feb 2)
    - Server – Charles
        - Choose Web Service or Simple approach (Jan 30)
        - Server able to communicate with client (Jan 30)
        - Add/remove users (Feb 2)
- Week 3: Feb 3 - Feb 9
    - UI – Kirsten
        - All basic controls completed (button, toolbar, chunk, listbox) (Feb 6)
        - Shapes listbox populated (oval, rectangle, triangle) (Feb 9)
        - Colors listbox populated (8 colors) (Feb 9)
        - Events notify DrawingBoard (tool change, color change, etc) (Feb 9)
    - User Interaction – Rob
        - Create shapes using mouse events (Feb 6)
        - Select shape by clicking (Feb 6)
        - Edit shapes using mouse events (move, resize) (Feb 9)
        - Implement free-form tool (Feb 9)
        - Implement text tool (Feb 9)
    - Client Backend – Sean
        - Communication between client and server working for entire grammar (Feb 9)
    - Server – Charles
        - Watches for user timeout (Feb 6)
        - Notifies all clients upon DrawingBoard change (Feb 9)
- Week 4: Feb 10 – Feb 16 - **Alpha Ready by end of week**
    - UI – Kirsten
        - Polish UI Appearance (Feb 13)
        - Testing (Feb 15)
        - Design user presence pane (Feb 16)
    - User Interaction – Rob
        - Apply color properties to objects (Feb 13)
        - Create “hook” for UI notification of active users (Feb 13)

        - Testing (Feb 16)
    - Client Backend – Sean
        - Optimize communication (Feb 13)
        - Testing (Feb 16)
    - Server – Charles
        - Network fault-tolerance checking (Feb 13)
        - Testing (Feb 16)
- Week 5: Feb 17 – Feb 23
    - UI – Kirsten
        - Testing and Integration (Feb 21)
    - User Interaction – Rob
        - Testing and Integration (Feb 21)
    - Client Backend – Sean
        - Testing and Integration (Feb 21)
    - Server – Charles
        - Testing and Integration (Feb 21)
    - Group – Client feedback
        - Meeting(s) to assess progress, new features, etc (Feb 23)
- Week 6: Feb 24 – Mar 1
    - UI – Kirsten
        - Design "locked" notification (shown when another user is editing) (Feb 26)
        - Design conflict resolution notification (shown when a user's changes are ignored) (Feb 28)
        - Keep user presence pane updated in real time (Mar 1)
    - User Interaction – Rob
        - Disable interactions with locked objects (Feb 27)
        - Design necessary "hooks" for UI notifications (Mar 1)
    - Client Backend – Sean
        - Build code to lock/unlock items using the server (Mar 1)
    - Server – Charles
        - Implement locking mechanism (Mar 1)
- Week 7: Mar 2 – Mar 8 – **Spring Break**
    - UI – Kirsten
        - Explore alternative conflict resolution scenarios (from user's perspective) (Mar 8)
        - Implement user login popup box
    - User Interaction – Rob
        - Plan how user may associate text with shapes (Mar 5)
        - Implement shape-text association (Mar 8)
    - Client Backend – Sean
        - Determine additional features (Mar 8)
    - Server – Charles
        - Determine server scalability (max clients, max operations per second, etc) (Mar 5)

- Week 8: Mar 9 – Mar 15 – **Beta Ready by end of week**
  - UI – Kirsten
    - Implement notifications (Mar 12)
    - Implement locking visual cue (Mar 15)
  - User Interaction – Rob
    - Polish user interaction (Mar 15)
  - Client Backend – Sean
    - Reserved for future features (Mar 15)
  - Server – Charles
    - Design layering algorithm for conflict resolution (Mar 15)
- Week 9: Mar 16 – Mar 22
  - UI – Kirsten
    - Add new chunk and buttons for layer-changing (Mar 20)
    - Add tooltips (Mar 22)
  - User Interaction – Rob
    - Resource loading asynchronously (Mar 20)
    - Show resource download progress on user login popup box (Mar 22)
  - Client Backend – Sean
    - Reserved for future features (Mar 22)
  - Server – Charles
    - Reserved for future features (Mar 22)
- Week 10: Mar 23 – Mar 29
  - UI – Kirsten
    - Add animations for user interface (Mar 29)
  - User Interaction – Rob
    - Reserved for future features (Mar 29)
  - Client Backend – Sean
    - Reserved for future features (Mar 29)
  - Server – Charles
    - Reserved for future features (Mar 29)
- Week 11: Mar 30 – Apr 5 – **Feature Freeze**
  - Everyone
    - Testing
    - Documentation
    - Polish, polish, polish
- Week 12: Apr 6 – Apr 12
  - Everyone
    - Project Video
    - Testing
    - Documentation
    - Polish, polish, polish
- Week 13: Apr 13 – Apr 19 – **Project Complete by end of week**
  - Everyone
    - Project Video

        - Testing
        - Documentation
        - Polish, polish, polish
- Week 14: Apr 20 – Apr 26
    - Everyone
        - Project Video (Apr 21)
        - Testing
        - Documentation
        - Polish, polish, polish

9. Glossary

.NET	The Microsoft .NET Framework is a component of Microsoft Windows operating systems. It includes a large library of solutions to common program requirements and tasks.
ASP.NET	The Microsoft ASP.NET Framework is a part of Microsoft's .NET platform. Like the .NET Framework, ASP .NET includes a large library of solutions to common program requirements and tasks; however, it is specifically targeted to aid in web programming.
C#	C# is an object oriented programming language that was developed by Microsoft as a part of .NET.
Client	A computer system that makes requests to a server.
IIS 7	Internet Information Services is a Microsoft Windows component that provides support for internet-based service. It is most well known for its use as a web server and can be used with ASP.NET to serve websites.
Serialization	The process of converting an object from memory to another format commonly with the intent of storing the object on long term storage or so that it may be transferred to another system.
Server	A computer system that is designed to respond to requests from other systems.
Session	A session contains all the information that is associated with a user that is visiting a website. The session is stored on the web server and the user's client maintains data necessary to inform the web server which session to use on each request. Sessions can be used to store all sorts of information but are usually used to allow a user to stay logged in to a website throughout his/her entire visit.
Silverlight	Microsoft Silverlight is a browser plugin that allows websites to embed Silverlight applications in web pages. These applications may contain features that traditional web pages do not support such as animations, audio-

	video playback, vector graphics, or other media-rich or interaction-heavy applications.
User presence	User presence is used to refer to the ability for one user to tell which other users are concurrently using the same application, in this case, the virtual whiteboard.
User Interface	The user interface of a computer program is the part of the program that users interact with directly. They typically include components such as buttons, labels, text, mouse cursors and scroll bars.
Visual Studio 2008	Microsoft Visual Studio 2008 is Microsoft's latest version of Visual Studio. Visual Studio is an application for programmers and aims to make development easier. It offers text-editing abilities as well as many additional tools such as a graphical debugger.
Windows Server 2003	Microsoft Server 2003 is a version of Microsoft Windows that has been customized to provide for the needs of web servers.
XAML	XAML stands for Extensible Application Markup Language and is an XML-based language often used to define visually rich user interfaces.