Chemistry Learning In Progress

Design Specifications Document

Nathan Mikeska
Neil Alfredson
Richard Carney
Brian Navarro

Table of Contents

1. Introduction

1.1 Purpose of the System

The purpose of this system is to provide students with an automated version of the current system. Therefore, this educational tool will have to retain the same objective of allowing a user to arrange freely a given set to form the patterns that make up the final arrangement. Each set in one way or another reflects the one of many ways the periodic table can be organized.

Furthermore, beyond mimicking the current system, we aim to provide observational data that shows the choices a user made toward finding a final arrangement for a set. This recorded data will help illuminate the various thought processes that occur for the user during their task of finding patterns within a set.

1.2 Design Goals

The primary design goals of this system are:

1.2.1 Dependability criteria:

Reliability – The system must accurately handle and display information to the end user.

Security – The system must be secured against unauthorized users modifying a user's local hard drive and it must properly inform the user of the need to read/write their hard drive.

Robustness – The system should perform error checking on all inputs in order to catch invalid inputs and deal with them in such a manner to prevent the program from crashing or disrupting the user's work. An example would be preventing the user from loading up a file for playback that is not actually a log file created by the system.

Availability – The system should be accessible to students and professors. It may also be used by other teachers and possible common non-related users.

1.2.2 Performance criteria:

Response time – The system should take less than one minute to download to the user's PC on a dialup connection. Once the

system is downloaded to the user's local PC, it should respond to all user input without any visible delay.

1.2.3 Maintenance criteria:

Modifiability – The system needs to be designed and implemented in an efficient object oriented manner in anticipation of any interface or functional application updates to the system.

Readability – The system layout, design, and code needs to be easy to navigate and understandable by any developer who would later be modifying the code or adding additional features. This can be accomplished by utilizing clear coding standards and module descriptions.

1.2.4 End User criteria:

Usability – The system should be usable to a wide range of users from novice to expert, and should be intuitive.

1.2.5 Design Tradeoffs

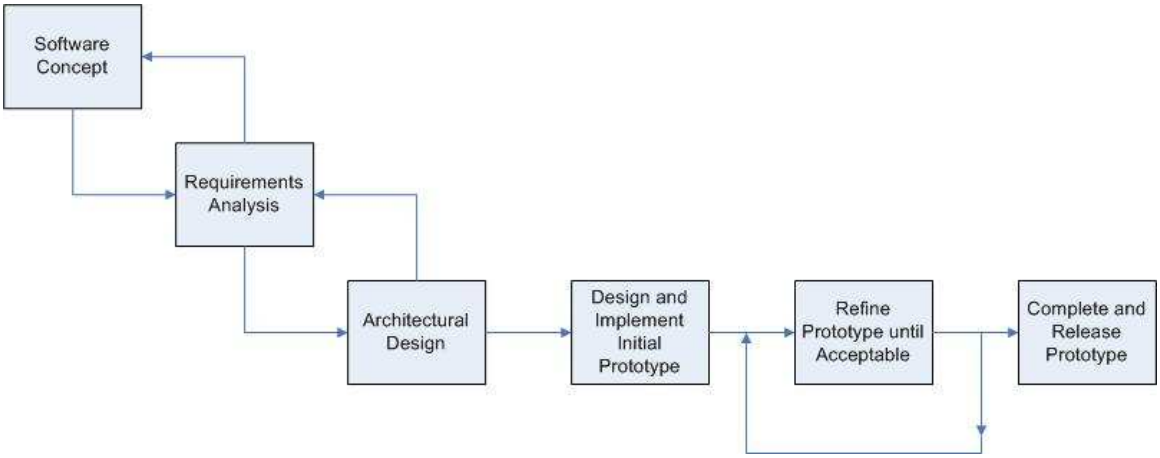
This project we as designers needed to consider several different design tradeoffs. A main tradeoff that we needed to consider and work around is tile size vs. screen size. We needed to keep the tiles large enough so that all of the information on each tile could be seen at one time. This caused some conflict because the larger and clearer the tiles were the less of them we could display on the screen at once. The size of the tiles also brought up conflicts with the size of the grid area. We needed the grid area to be large enough to accommodate the patterns of the set. The patterns could be several tiles long or high. We could not afford to make the tiles smaller so we needed to add the functionality of the scrolling grid area to accommodate all different kinds of patterns. The tradeoff to make the grid larger then can be displayed at once forced us to trade the simplicity of our program and add a way to see the entire pattern at once by means of a mini-map.

The use of java is another design tradeoff that our group needed to make. We needed to consider the difference between functionality and the learning curve associated with learning the new language. The time that it takes to learn java was weighed against the need for additional functionality and the deadline we

had for our project. After deciding to use the java language we had to determine the pros and cons of using an applet vs. application. The applets ability to be online-based and used from a web browser made us try to develop with applet in mind. However, do to the need to sign the applets in order to read and write to files we needed to change our program so it would be an application. The requirement that the program be available online now means that they need to download the application from a website.

Security was a main consideration of the group. We needed to determine who we should allow to use the system and what we would allow them to do. The security of the system was sacrificed in order to make the program available and ease of use. We decided that instead of trying to keep the application secure by requiring a logon or some password we would allow all users to use the program. We also decided that we would allow the students and teachers to have the same access to the functionality of the program. This allowed us to only worry about developing a single interface for all users. Another security tradeoff that we needed to worry about was the submission of the results. We could have allowed the program to submit all the results to the professor, however this could lead to many complications for the professor due to users of the system that are not in her class. This would also complicit allowing other professors to use the system. So we decided that it would be better to have the submission of the results to the professor separate from the program. This means that the student could email the recoded file to the professor or give it to her on a disk. Another option that was explored was the use of Blackboard to turn the assignment into the teacher. The teacher could also tell the students that they need to print out a copy of there final results and turn in the printout. The printing functionality in the program allows the user to print tile arrangements, print additional tile information, or print to image files.

1.2.6 Lifecycle Model



The lifecycle model that we will use for the entirety of project will be the waterfall model. However, within our coding and testing phases, we will gear our model to a more evolutionary prototyping approach.

Given the small team size and project milestone deadlines, the waterfall model provides a solid lifecycle that we can follow without needing to make compromises to our goals or deadlines. During the coding and testing phases of the lifecycle, our project will take an evolutionary prototyping approach. It is very important that the system be easily understood and used by the users. This will require a strong focus on providing a good user interface for users to interact with the system.

Therefore, the evolutionary prototyping will allow us to gather important feedback on each iteration of the prototype so that we may provide the best possible interface for the users.

1.3 Definitions, acronyms, abbreviations

C.L.I.P - Chemistry Learning In Progress
Professor (teacher) – A user that accesses the system from the perspective of a certain role. This role includes the use of the playback and creation functionality.
Student – A user that accesses the system from the perspective of a certain role. This role includes the use of the tile arrangement functionality.
User – Every user has access to the entire functionality of the system. Generally a user fulfills the role of a professor or student as defined above.

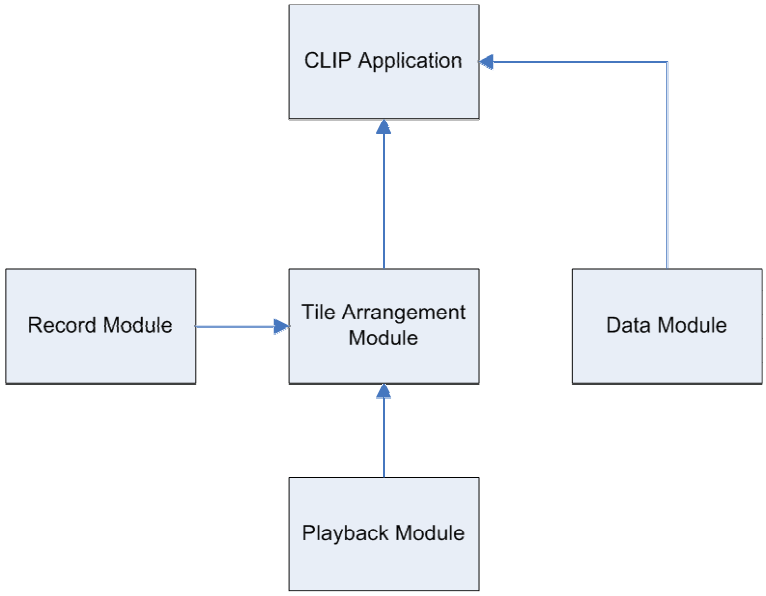
1.4 References

Our Website
http://www.cs.siue.edu/seniorprojects/2005/fall/CLIP/

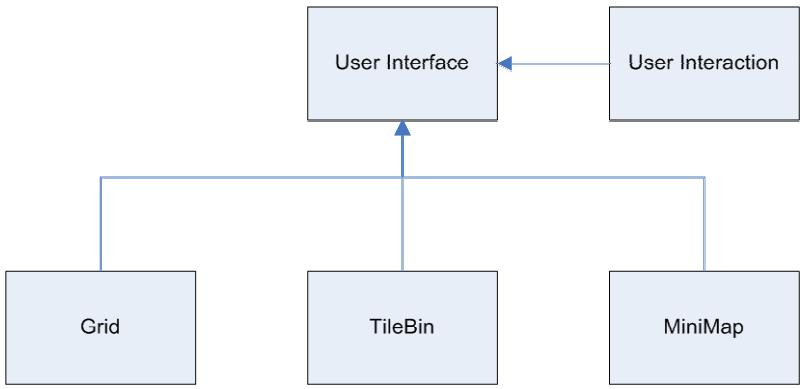
2. Proposed System Architecture

2.1 System Overview

Complete System



Tile Arrangement Module



2.2 System Decomposition

There are 4 main modules that make up our system. They are the Tile Arrangement module, Record module, Playback module, and Data module.

Tile Arrangement Module

The Tile Arrangement module can be broken down into 5 main components: the User Interface, User Interaction, Grid, TileBin, and MiniMap. The User Interface contains the Grid, TileBin, and MiniMap as well as the other components that make up the overall user interface of the application. The User Interaction component handles all interactions the user makes with the interface, Grid, TileBin, and MiniMap. The Grid component is the grid in which tiles are arranged upon. The TileBin component holds all tiles not currently on the Grid. The MiniMap provides an overview of the Grid as well as a convenient way to move around in the Grid.

Record Module

The Record module logs all moves and significant events that occur during the arrangement of tiles. If and when the user decides to save his or her arrangement, the Record module handles the saving of the log file.

Playback Module

The Playback module is responsible for loading up and allowing the user to view log files. The Playback module implements a control panel with various controls used to examine the log. It also has the ability to auto-playback logs.

Data Module

The Data module contains the interface and functionality needed for the user to create or modify tile sets.

2.3 Hardware/Software Mapping

The CLIP system is available for download online and requires that the user have Java. A .jar file will hold the application and will be stored on a server. Once downloaded, the application can be executed and used on the user's local PC.

2.4 Persistent Data Management

The Clip system will utilize the following files:

Log file

The log file will store the list of moves made by the user. During playback, this file will be read by the system in order to play back the user's moves. The log files also include the entire tile set associated with them.

Rule Set file

The rule set file will store rules and options that define a tile set. Information to be stored consist of set name, set description, tile size, ability to add a blank tile, and randomly remove tile. The rule set files also include information about the grid size, overall comments, and unused tile information. By using serializable class it also holds all of the images and information about each of the tiles.

Image files

The image files are the GIF, PNG, or JPG images used to represent the tiles.

2.5 Access Control and Security

CLIP is an online system that is accessible to anyone with an internet connection and java. There will be no restriction on the systems functionality regardless of the user.

2.6 Global Software Control

The system is event driven. The actions taken by the users either by dragging/dropping a tile, pressing a command button, or selecting menu option, will determine which sub system is activated. If there is no action taken by the user then system is idle.

2.7 Boundary Conditions

Initialization

The system will be a java application. There will downloadable jar file to be hosted at a location of our client's choice and accessible to users. It will allow the user to save their work as individual files to their own computer and also allow users to view the playback given a saved file.

Termination

The system will be shut down by exiting the application.

Failure

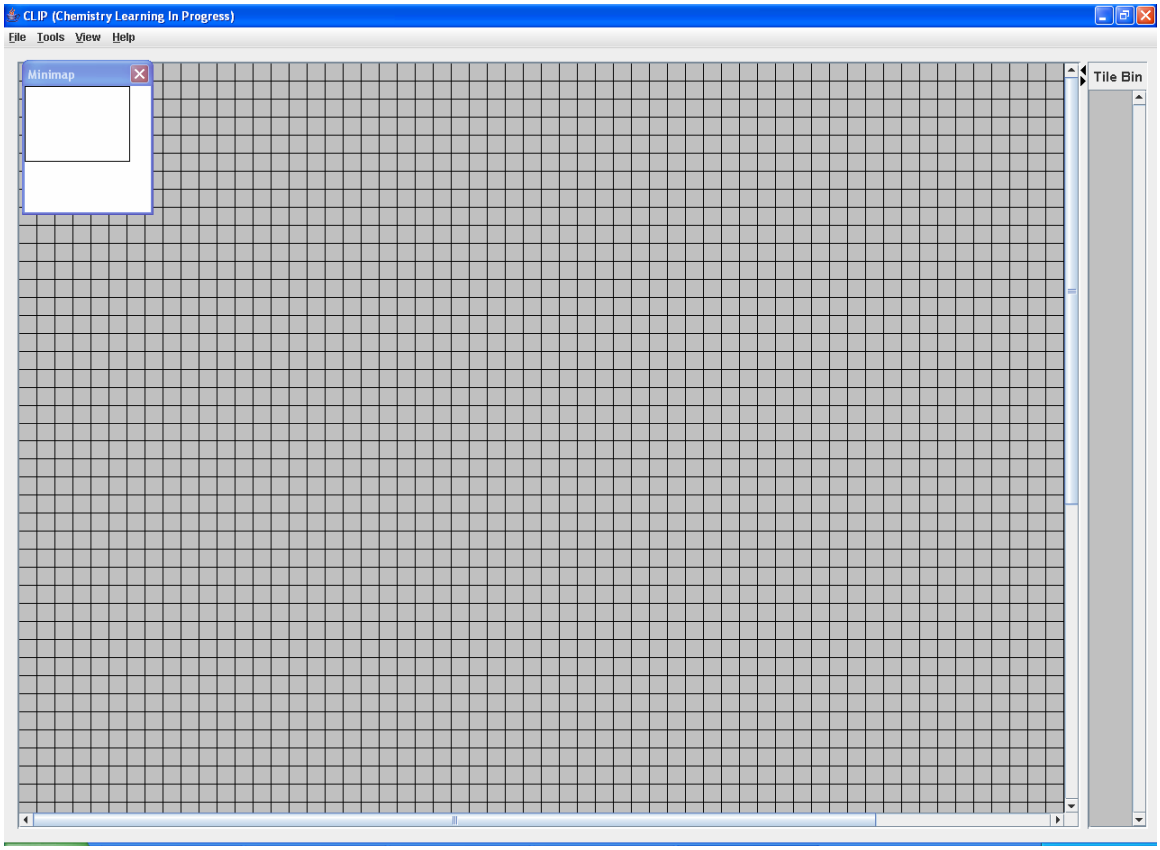
After an unexpected client side failure, the system will display a brief, descriptive error message to the user before termination.

3. User Interface

The User Interface of the C.L.I.P system is important in recreating the look and feel of current system's physical environment. Furthermore, our system also focuses on providing ways to easily create/modify tile sets, load existing tile sets up for use in tile arrangement, and allow for easy play back of previously done tile arrangements. There is also the ability to print information.

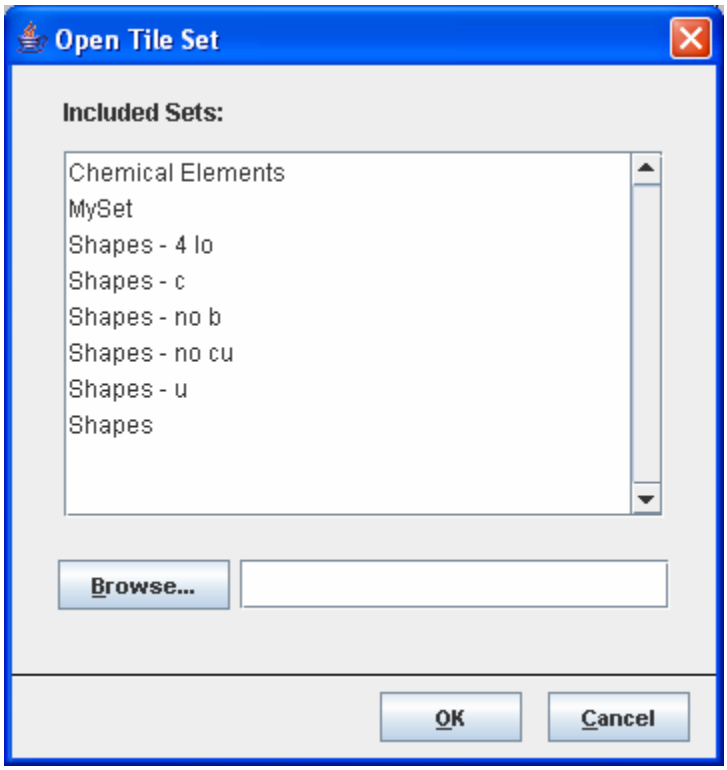
Main Screen – Initial Look

After the application has successfully loaded, the main screen will appear. This screen will contain a menubar, an empty grid, the minimap, and an empty tile listing.



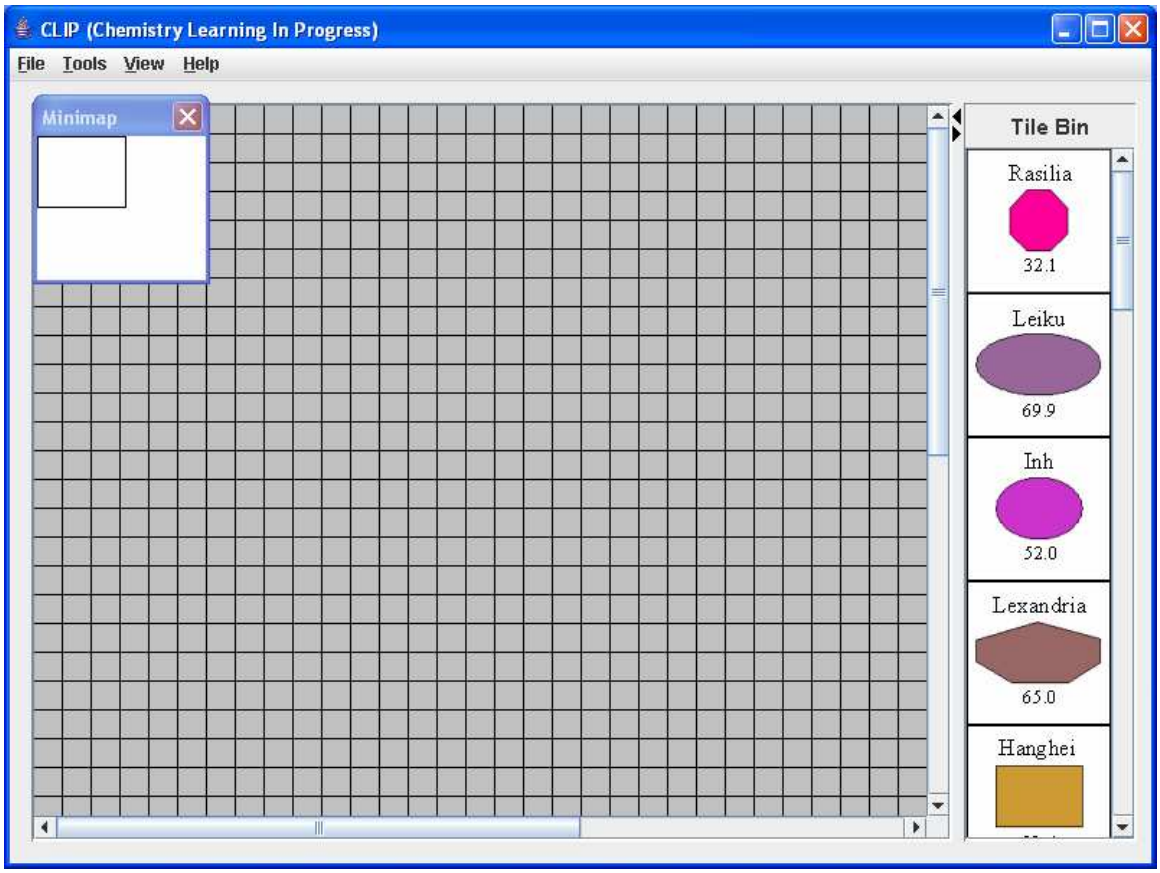
Open Tile Set Dialog

In the file menu, a user will select open and choose tile set after which an open tile set dialog will appear. The dialog will display a listing of preloaded sets. These are the sets that are available already with the application. The user also has the option to browse for additional sets that may be located elsewhere such as on the computer's hard drive.



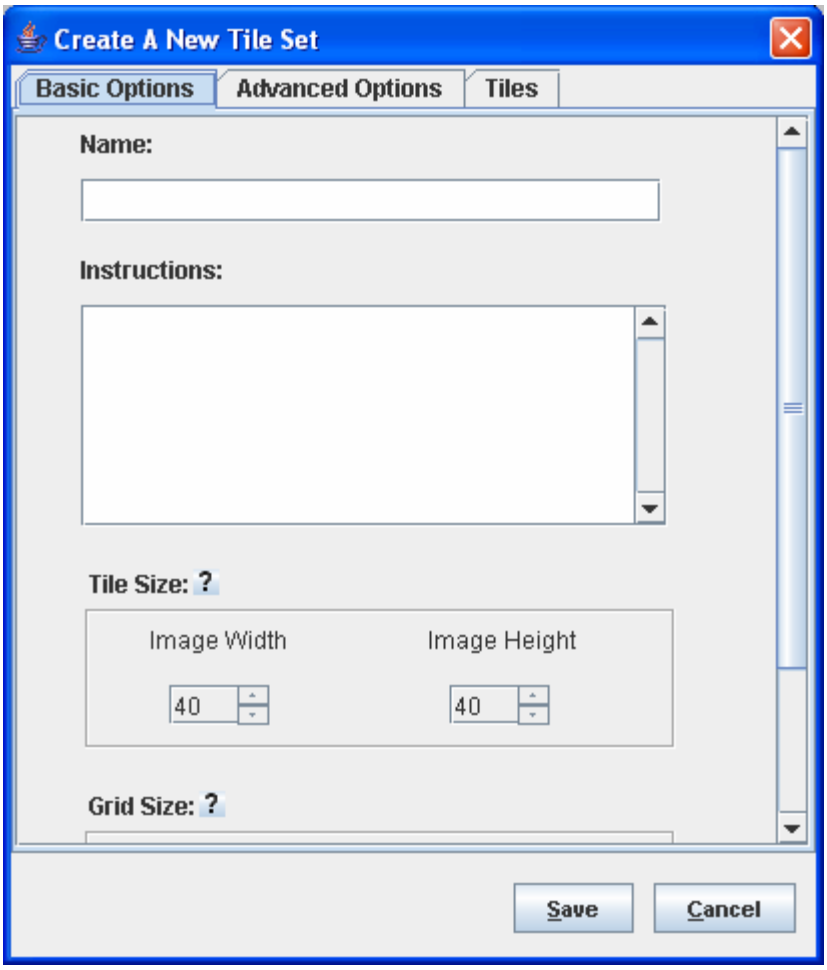
Main Screen – Tile Set Loaded

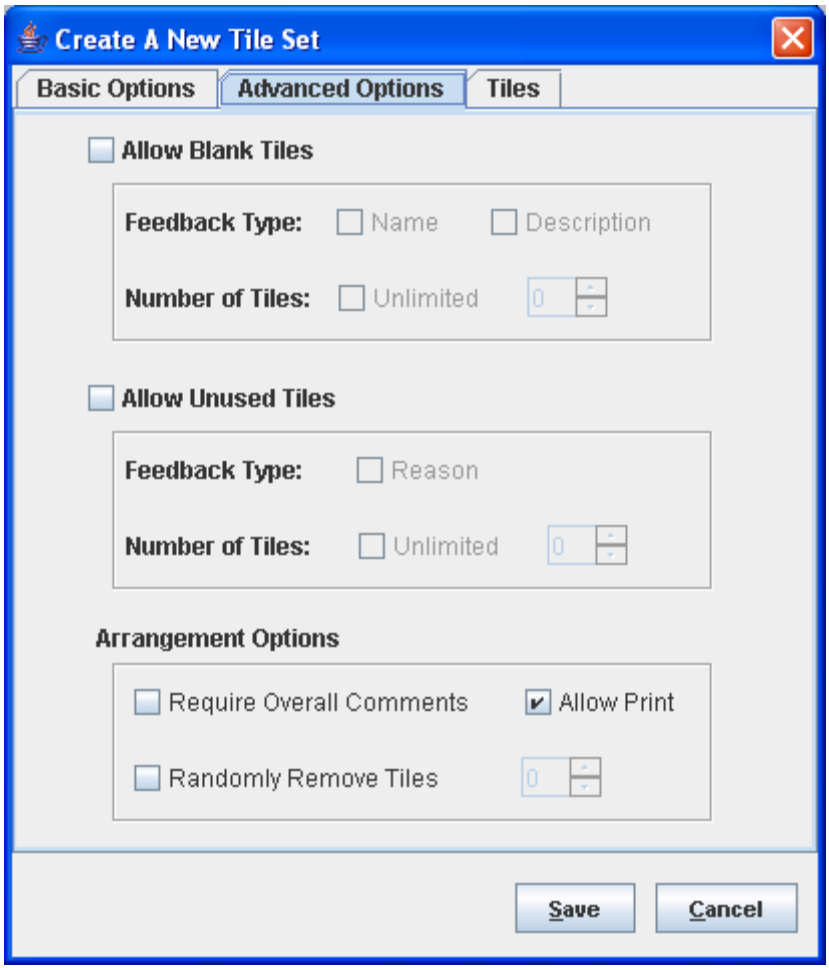
After the user has selected a tile set, its contents will be loaded into the tile listing to the right of the grid. The order of the tiles will be random and different each time a set is loaded.

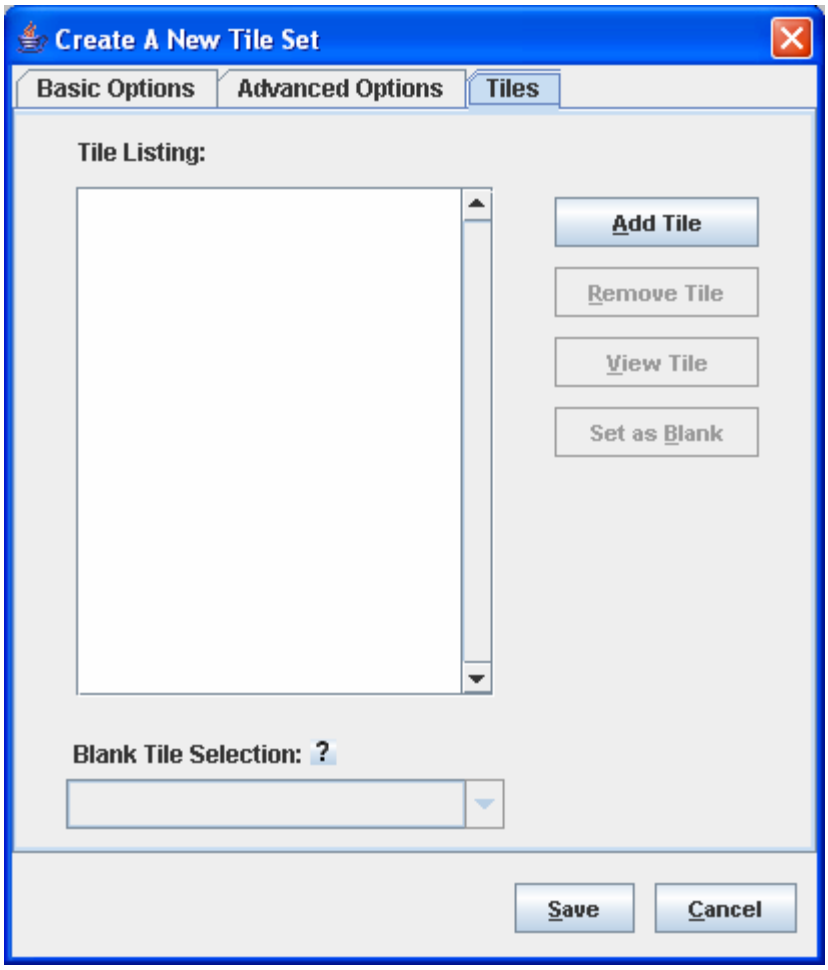


Create Set – Rule Editor, Tile Editor

When a user wishes to create a new set, they can select from the menu bar, Tools -> Create Set. A Create Set dialog will appear with 3 tabs. In Basic Options, the user needs to provide information about what the name of the set will be, the instructions of the set, the size of the tiles the set will use, and the size that they want the grid to be. In Advanced Options, the user chooses what options they want the set to use including options for blank tiles, unused tiles, randomly removing tiles. The Tiles tab allows a user to start adding tile images to the set they defined under the rule editor. The rule editor outlines all the characteristics that define a certain set.

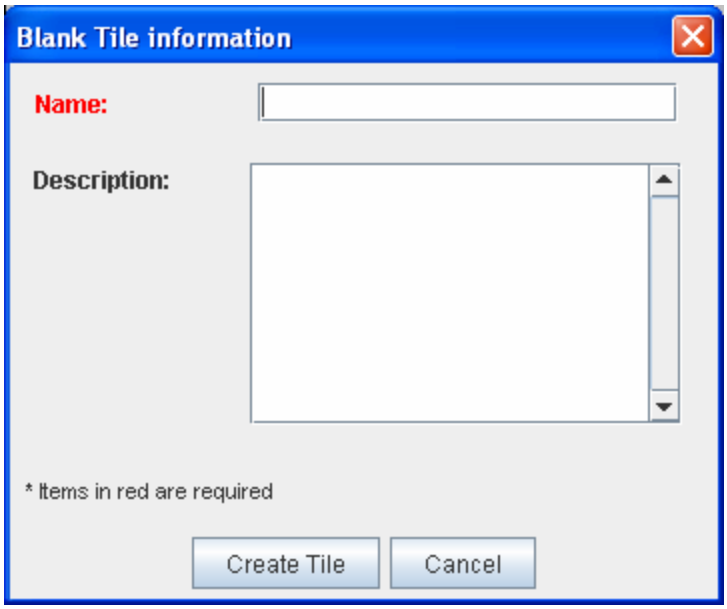

Create A New Tile Set
Basic Options
Advanced Options
Tiles
Name:
Instructions:
Tile Size: ?
Image Width
Image Height
40
40
Grid Size: ?
Save
Cancel


Create A New Tile Set
Basic Options
Advanced Options
Tiles
Allow Blank Tiles
Feedback Type:
Name
Description
Number of Tiles:
Unlimited
0
Allow Unused Tiles
Feedback Type:
Reason
Number of Tiles:
Unlimited
0
Arrangement Options
Require Overall Comments
Allow Print
Randomly Remove Tiles
0
Save
Cancel



Insert Blank Tile

For some sets it is allowable to add blank tiles that take the place of tiles that are not in the set, but that the user believes need to be in order to fill in the gaps in their arrangement. To add a blank tile, a user will select from the menu bar Tools-> Add Blank Tile and a dialog with appear. In this dialog, the user will describe the characteristics of the missing tile. The user can also right click and get the option to add a blank tile from a popup context menu. The red fields are those that are required based on the rules of the set.



4. Subsystem Interfaces

Tile Arrangement Module

The key line of communication in the Tile Arrangement module is the User Interaction component. The User Interaction component handles all communication between all the components that make up the Tile Arrangement module. This means that the Grid, TileBin, and MiniMap don't need to directly communicate with each other. Externally, the User Interaction component sends information to the Record module to be recorded and later saved as a log file.

Record Module

The Record module receives information from the User Interaction module. Other than receive information from the Tile Arrangement module, this module has no need to communicate or send information to other modules.

Playback Module

This module sends information to the Tile Arrangement module. The information being sent is commands to do specific things related to the current log entry being executed.

Data Module

The Data module has no need to communicate with the other modules since its only purpose is to create and modify tile sets.

5. Packages and File Organization

Set Files

The set files are sets of picture files that are to be loaded by a user to be arranged on the grid. The system will begin with several different sets of tiles for use by the users. To allow the expansion of the program the system can have additional tile sets created. The set file will store a list of tiles. This file will be created from create set item in the tools menu. The rule set file has all of the information about a specific set. It has all of the options that are enabled or disabled in the set, such as the ability to leave out tiles, add blank tiles, and randomly remove tile. This file will also store the information about tile size, the name of the set, and the description and rules associated with the set. This file will be made when a user creates a new set. This is through the create set option in the tools menu. The user selects the option that should be enabled in the new set and the name and description of the set. This file can be edited by the modify set option in the tools menu.

Log Files

The log file stores information about all of the moves that were made including all of the blank tiles that were added while the user was using the program. This file will be created when the user is in arrangement mode and saves a file. The user can select two different save options Complete Tile Arrangement or Incomplete Tile Arrangement. If Complete Tile Arrangement is chosen the user could be presented with some dialog boxes where the user can provide description for their arrangement and unused tiles before the save file dialog box pops up. The menu will pop up a save file dialog box and allow the user to save the file to the local hard drive. This file will also be used in the Playback mode for file reading. The file will be accessed when the user selects the Open and chooses Tile Arrangement. The program will read all of the moves that are saved on the file and display them one at a time. All entries will be time-stamped with the time (hours:minutes:seconds) since the last action. The user can also use log files to continue from where they left off by choosing Continue Tile Arrangement from the open menu.

The file will have 12 different moves:

TILEBIN_TO_TILEBIN
TILEBIN_TO_GRID

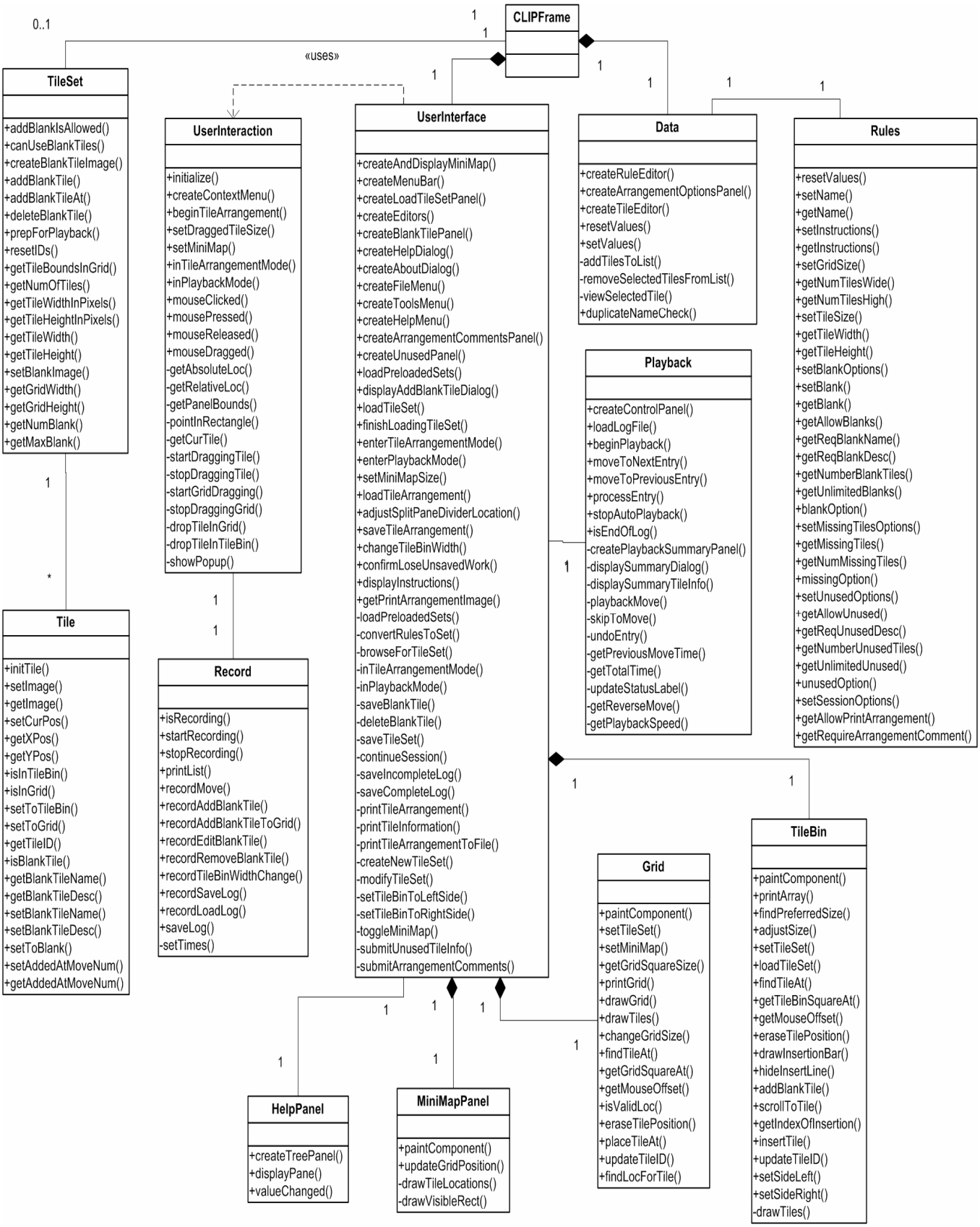
GRID_TO_GRID
GRID_TO_TILEBIN
ADD_BLANK_TILE
EDIT_BLANK_TILE
REMOVE_BLANK_FROM_GRID
REMVOE_BLANK_FROM_TILEBIN
CHANGE_TILEBIN_WIDTH
ADD_BLANK_TILE_TO_GRID
SAVE_LOG
LOAD_LOG

TILEBIN_TO_TILEBIN when a tile moves from the tile bin to a different place in the tile bin. TILEBIN_TO_GRID when a tile moves from the tile bin to the grid area. GRID_TO_GRID when a tile moves from the grid to a different position on the grid. GRID_TO_TILEBIN when a tile is moved from the grid to the tile bin. ADD_BLANK_TILE when the user has added a blank tile. EDIT_BLANK_TILE when the user has edited a blank tile. REMOVE_BLANK_FROM_GRID and REMOVE_BLANK_FROM_TILEBIN is when the user deletes one of the blank tiles that had been added. CHANGE_TILEBIN_WIDTH when the user has changed the size of the tile bin. ADD_BLANK_TILE_TO_GRID is what happens when the user adds a blank tile to the grid area via context menu. SAVE_LOG is when the user saves the arrangement that they currently have. LOAD_LOG happens when the user loads up an existing file in arrangement mode.

Image Files

The image files are the *.png, *.jpg, or *.gif files that have the image of tiles on them. These are what will be used to display the information for the tile to the user. The user must create or find their own files for this purpose.

6. Class Diagram


CLIPFrame
«uses»
0..1
1
TileSet
+addBlankIsAllowed()
+canUseBlankTiles()
+createBlankTileImage()
+addBlankTile()
+addBlankTileAt()
+deleteBlankTile()
+prepForPlayback()
+resetIDs()
+getTileBoundsInGrid()
+getNumOfTiles()
+getTileWidthInPixels()
+getTileHeightInPixels()
+getTileWidth()
+getTileHeight()
+setBlankImage()
+getGridWidth()
+getGridHeight()
+getNumBlank()
+getMaxBlank()
*
Tile
+initTile()
+setImage()
+getImage()
+setCurPos()
+getXPos()
+getYPos()
+isInTileBin()
+isInGrid()
+setToTileBin()
+setToGrid()
+getTileID()
+isBlankTile()
+getBlankTileName()
+getBlankTileDesc()
+setBlankTileName()
+setBlankTileDesc()
+setToBlank()
+setAddedAtMoveNum()
+getAddedAtMoveNum()
UserInteraction
+initialize()
+createContextMenu()
+beginTileArrangement()
+setDraggedTileSize()
+setMiniMap()
+inTileArrangementMode()
+inPlaybackMode()
+mouseClicked()
+mousePressed()
+mouseReleased()
+mouseDragged()
-getAbsoluteLoc()
-getRelativeLoc()
-getPanelBounds()
-pointInRectangle()
-getCurTile()
-startDraggingTile()
-stopDraggingTile()
-startGridDragging()
-stopDraggingGrid()
-dropTileInGrid()
-dropTileInTileBin()
-showPopup()
Record
+isRecording()
+startRecording()
+stopRecording()
+printList()
+recordMove()
+recordAddBlankTile()
+recordAddBlankTileToGrid()
+recordEditBlankTile()
+recordRemoveBlankTile()
+recordTileBinWidthChange()
+recordSaveLog()
+recordLoadLog()
+saveLog()
-setTimes()
UserInterface
+createAndDisplayMiniMap()
+createMenuBar()
+createLoadTileSetPanel()
+createEditors()
+createBlankTilePanel()
+createHelpDialog()
+createAboutDialog()
+createFileMenu()
+createToolsMenu()
+createHelpMenu()
+createArrangementCommentsPanel()
+createUnusedPanel()
+loadPreloadedSets()
+displayAddBlankTileDialog()
+loadTileSet()
+finishLoadingTileSet()
+enterTileArrangementMode()
+enterPlaybackMode()
+setMiniMapSize()
+loadTileArrangement()
+adjustSplitPaneDividerLocation()
+saveTileArrangement()
+changeTileBinWidth()
+confirmLoseUnsavedWork()
+displayInstructions()
+getPrintArrangementImage()
-loadPreloadedSets()
-convertRulesToSet()
-browseForTileSet()
-inTileArrangementMode()
-inPlaybackMode()
-saveBlankTile()
-deleteBlankTile()
-saveTileSet()
-continueSession()
-saveIncompleteLog()
-saveCompleteLog()
-printTileArrangement()
-printTileInformation()
-printTileArrangementToFile()
-createNewTileSet()
-modifyTileSet()
-setTileBinToLeftSide()
-setTileBinToRightSide()
-toggleMiniMap()
-submitUnusedTileInfo()
-submitArrangementComments()
Data
+createRuleEditor()
+createArrangementOptionsPanel()
+createTileEditor()
+resetValues()
+setValues()
-addTilesToList()
-removeSelectedTilesFromList()
-viewSelectedTile()
+duplicateNameCheck()
Playback
+createControlPanel()
+loadLogFile()
+beginPlayback()
+moveToNextEntry()
+moveToPreviousEntry()
+processEntry()
+stopAutoPlayback()
+isEndOfLog()
-createPlaybackSummaryPanel()
-displaySummaryDialog()
-displaySummaryTileInfo()
-playbackMove()
-skipToMove()
-undoEntry()
-getPreviousMoveTime()
-getTotalTime()
-updateStatusLabel()
-getReverseMove()
-getPlaybackSpeed()
Rules
+resetValues()
+setName()
+getName()
+setInstructions()
+getInstructions()
+setGridSize()
+getNumTilesWide()
+getNumTilesHigh()
+setTileSize()
+getTileWidth()
+getTileHeight()
+setBlankOptions()
+setBlank()
+getBlank()
+getAllowBlanks()
+getReqBlankName()
+getReqBlankDesc()
+getNumberBlankTiles()
+getUnlimitedBlanks()
+blankOption()
+setMissingTilesOptions()
+getMissingTiles()
+getNumMissingTiles()
+missingOption()
+setUnusedOptions()
+getAllowUnused()
+getReqUnusedDesc()
+getNumberUnusedTiles()
+getUnlimitedUnused()
+unusedOption()
+setSessionOptions()
+getAllowPrintArrangement()
+getRequireArrangementComment()
Grid
+paintComponent()
+setTileSet()
+setMiniMap()
+getGridSquareSize()
+printGrid()
+drawGrid()
+drawTiles()
+changeGridSize()
+findTileAt()
+getGridSquareAt()
+getMouseOffset()
+isValidLoc()
+eraseTilePosition()
+placeTileAt()
+updateTileID()
+findLocForTile()
TileBin
+paintComponent()
+printArray()
+findPreferredSize()
+adjustSize()
+setTileSet()
+loadTileSet()
+findTileAt()
+getTileBinSquareAt()
+getMouseOffset()
+eraseTilePosition()
+drawInsertionBar()
+hideInsertLine()
+addBlankTile()
+scrollToTile()
+getIndexOfInsertion()
+insertTile()
+updateTileID()
+setSideLeft()
+setSideRight()
-drawTiles()
HelpPanel
+createTreePanel()
+displayPane()
+valueChanged()
MiniMapPanel
+paintComponent()
+updateGridPosition()
-drawTileLocations()
-drawVisibleRect()

7. Testing

Module Testing

As detailed by the lifecycle model our coding and testing phases will follow an evolutionary prototyping approach. Therefore code will be written onto the latest approved version of the prototype. This eliminates the need for imitation drivers or stubs as the modules completed will be tested using the latest accepted version of the prototype. An example of this would be the testing of the playback module.

Integration Testing

Once members have finished coding and testing the individual modules that they are working on, the code will be written into the main prototype. Even though members will be using their own copy of latest version of the prototype while writing their code, it is not unreasonable to expect that different modules will be coded by different team members concurrently. Therefore this testing is to assure that the code being written into the main prototype is compatible with another member's code and previously coded modules. An example of this would be testing the integration between the grid, tile arrangement, and playarea modules by moving a tile from the play area and placing it in the grid. The result would be immediately visible on the screen. The result can also be checked by looking at the log file created if the save arrangement button is clicked after the move.

System Testing

In our evolutionary prototyping development the system test will be a complete test over the systems expected functionality with the modules it contains at the time of testing. Since module and integration testing will be performed first, system testing is the last test to be performed before the prototype is advanced to the next acceptable version. This will be the version that the members will use as the driver to their new modules. The actual testing will consist of a repetition of all the module testing and integration testing that has taken place between version upgrades. System testing has been performed numerous times and the complete system is then checked for more bugs do to conflicts in the code.

Acceptance Testing

After most of the functionality had been added to the system we began acceptance testing. This testing the team observed 14 selected students using the system in the HCI lab. This helped us refine our system and come up with better interface.

Task Oriented

During these tests the user was seated next to a team member in front of a computer running the system. The other team members will be in the other room of the HCI lab operating the cameras, VCR, and observing the test. The team member leading the test and next to user will be asking the user to perform a series of specific tasks on the system

These tasks to be performed are:

- open the application
- load the first tile set into the tile bin
- reorganize the play area by taking the third tile and placing it in the first spot
- move all of the tiles to any locations on the grid
- arrange the tiles on the grid in the most logical arrangement according to the student
- move a specific tile from the grid to the tile bin
- add a blank tile to the set of tiles
- save the final arrangement of the grid
- fill in the pop up form asking why they left the tile placed back into the tile bin out of the final arrangement
- open the recording the user submitted earlier
- move ten steps forward
- move a step backward
- auto play the rest of the moves till the last recorded move
- modify a tile set by adding another tile
- modify the same tile set by removing a different tile from the tile set
- create a new tile set from images preloaded onto the computer
- create a new best arrangement pattern for the new tile set
- adjust the rules of the new tile set to not allow students to input blank tiles
- adjust the rules of the new tile set to not allow students to add new best arrangement patters
- turn off the gridlines

During these sessions will be focusing primarily on the use of the system through the perspective of the student. However, to fully test all aspects of our system we had the students perform the tasks that relate to the teachers role such as the playback of arrangements and tile set modifications. The team will record any mistakes the user makes, as well as their response as to why they made the mistake, while performing the tasks. Examples of mistakes may include clicking on the wrong menus or moving the tiles to wrong locations. We will also be keeping track of the time it takes the student to complete the task. From these tests we hope to discover any problems with interface that prevents the students from accessing the functionality of the system.

Acceptance Testing II
User: Chemistry Professor

The tasks to be performed are:
- the same tasks as students

These sessions will primarily focus on the use of the system through the role of the teacher. However they will need to complete the tasks the students have, because their understanding of the systems is more complex. The teachers will be evaluated and observed the same way as the students. During these tests we hope to locate any misunderstandings the teachers may have due to the interface organization.

<u>Open Environment</u>

These tests were conducted similar to tests above. However the user was not be asked to perform or be guided through specific tasks. They were given a brief description on the purpose of the system and be asked to simply use it. While they are guiding their way through the system the lead tester asking them questions about the language of the menu options, size of the forms, difficulty of scrolling, difficulty of using the mini-map, colors of the forms, visibility of the information on the tiles, organization of the menus, and overall comfort with using the system. From these tests we

hoped to discover anything in the interface that is unappealing to the users.

Online Testing

In addition to all of the user sessions that we are doing we also had an online version of our current working prototype that is available for anyone to use. We then gave visitors to the website a series of tasks and several different questioners. One of the benefits of doing this was to get professors that didn't have time to go to user sessions feedback about our program. This was important in helping us get as much feedback as we could, including information about users of different platforms and settings. Many of the tasks that were asked of the online participants were the same as those for the user testing.

8. Glossary

Version #	Date	Author	Description
0.1	10/05/05	Brian Navarro	User Interface Design, Research Topics
	10/06/05	Neil Alfredson	Purpose of the System, Design Goals (Except for Lifecycle model), Boundary Conditions, Packages and File Organization
	10/07/05	Nathan Mikeska	Lifecycle model, System Decomposition, and System Overview
0.2	10/08/05	Richard Carney	Global Software Control, Access Control and Security, Persistent Data Management, Hardware/Software Mapping. Revised the System Overview.
	10/08/05	Brian Navarro	More User Interface Design, Web Page Content Description, and Research Topics.
	10/08/05	Nathan Mikeska	System Decomposition (Revised), Subsystem Interfaces, Class Diagrams.
	10/08/05	Neil Alfredson	Testing
0.3	10/09/05	Brian Navarro	Design Specification Document Formatting, References
0.4	10/09/05	Nathan Mikeska	Class Diagram
1.0	10/09/05	All Members	Minor grammar and spelling fixes. Reviewed by team and accepted as version 1.0
1.0 - 1.3	10/09/05 12/07/05	All Members	Thorough revision of entire document. Most sections updated
2.1	4/22/06	Neil Alfredson	Updated document from first semester to bring it up to date with the final product.
2.2	4/28/06	Nathan Mikeska	Minor updates and changes.