Documentation

Created by

Supakorn Kijwattanachai 6232030521

2110215 Programming Methodology Semester 2 Year 2019 Chulalongkorn University

Chip on the grid

Introduction

Chip on the grid is inspired by one of Codeforces problem. https://codeforces.com/gym/102267/problem/E . It's puzzle problem where the input is 4 initial locations of the chips. Try to move all chips to the goal within 1000 moves. The puzzle was really interesting, but I was tired because it was very hard to solve puzzle on the paper. That's why I want to create helper tool to solve this kind of problems.

Game Features

This game has two main features, playing standard map and creating map.

Playing standard map feature

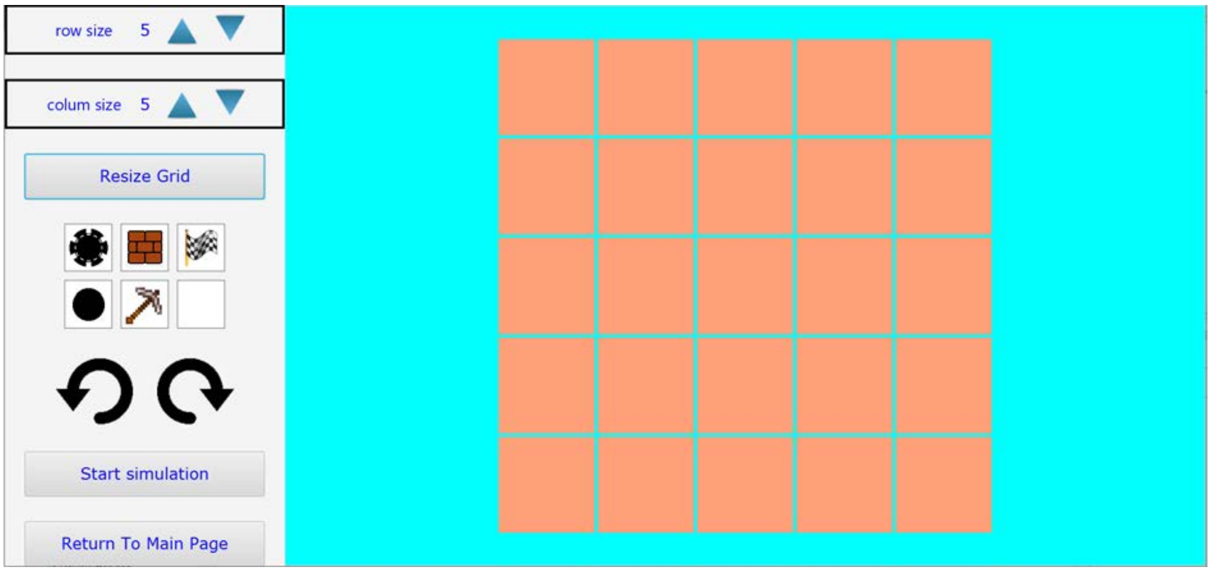
Game's rule for standard map

1) You can move chips by pressing ctrl+arrow and all of the chips will move respected to arrow direction. (All chips move simultaneously)
2) In standard map, the number of goals is equal to number of chips, your winning condition is to move all chips to goal.
3) If the cell is occupied by block or other chips, the chip cannot move into that cell.
4) If the cell is occupied by hole, the game is over.

You can undo and redo as many as you wants, but note that the game over states will not be saved.

Creating map feature/Simulation scene

The general purpose of this feature is to allow you to simulating the process of moving chips on the grid.



The features all as following.

1) **Row size bar**, to customize number of grid's row.
2) **Column size bar,** to customize number of grid's column.
   In these two bars, you can press up triangle to increase number by 1 and down triangle to decrease number by 1. The minimum allow is 1 and maximum allow is 20.
3) After you finish customize row size and column size, you can press **resize grid's** button to make new grid. The data for each cell from previous grid will be transfer to new grid if the location from previous grid can be located in new grid.
4) **Item pane**. There are 5 objects, chip, block, goal flag, hole, and destroy tools.
   You can select items (Chip, block, flag, hole) and click on the cell of right side grid to put it in. You will successfully put the item if it doesn't break game rules.
   You can select destroy tool and click on the cell of right side grid to clear the cell.
5) **Undo and redo buttons**. When you move chips, the moves will be recorded, you can press counter-clockwise arrow to undo and clockwise arrow to redo. But be careful, if you successfully place object, resize grid, or quit this page, the history will be all gone.
6) **Start simulation button**. You can't move unless you click this button. After you clicked, it will display "Stop simulation" instead. During simulation, you cannot resize grid or place the object. If you click "Stop

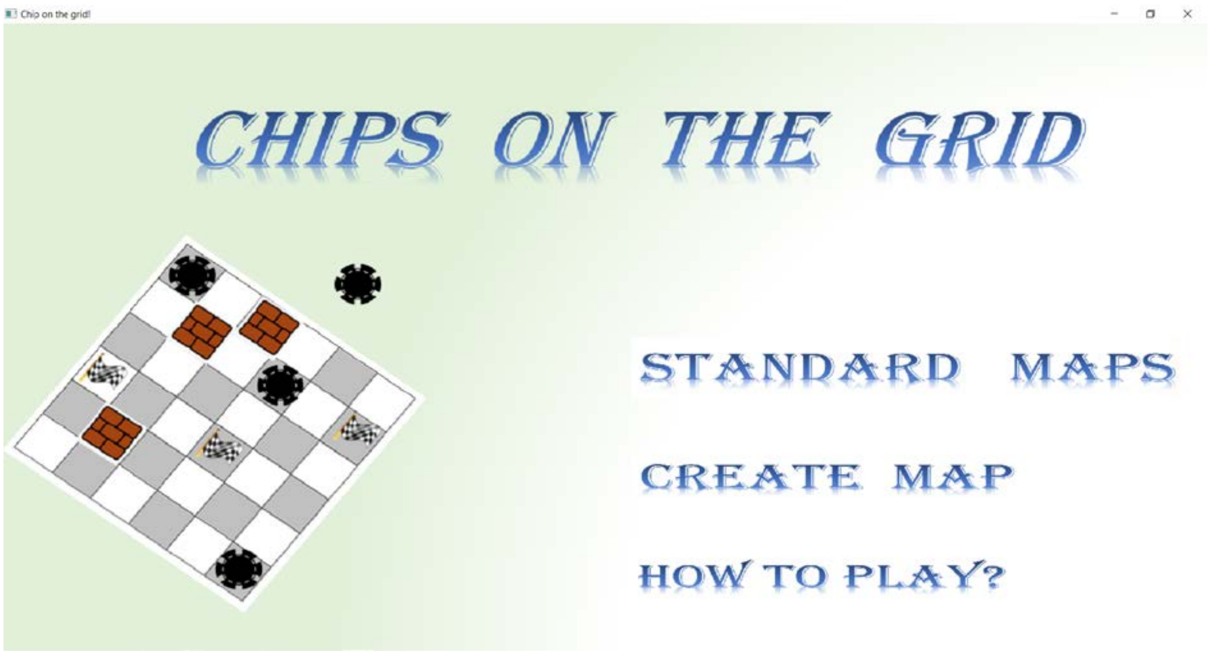
simulation", it will turn back to "Start simulation" and you cannot move again, but you are allowed to place object and resize grid.

7) **Return to Main Page button**. The name itself clearly tells you that you will go back to the main page.

The rule for creating map features will be little different.

The number of chips and goal isn't necessary to be the same. The winning condition is to move all chips into goal and number of goal > 0.

Main menu scene



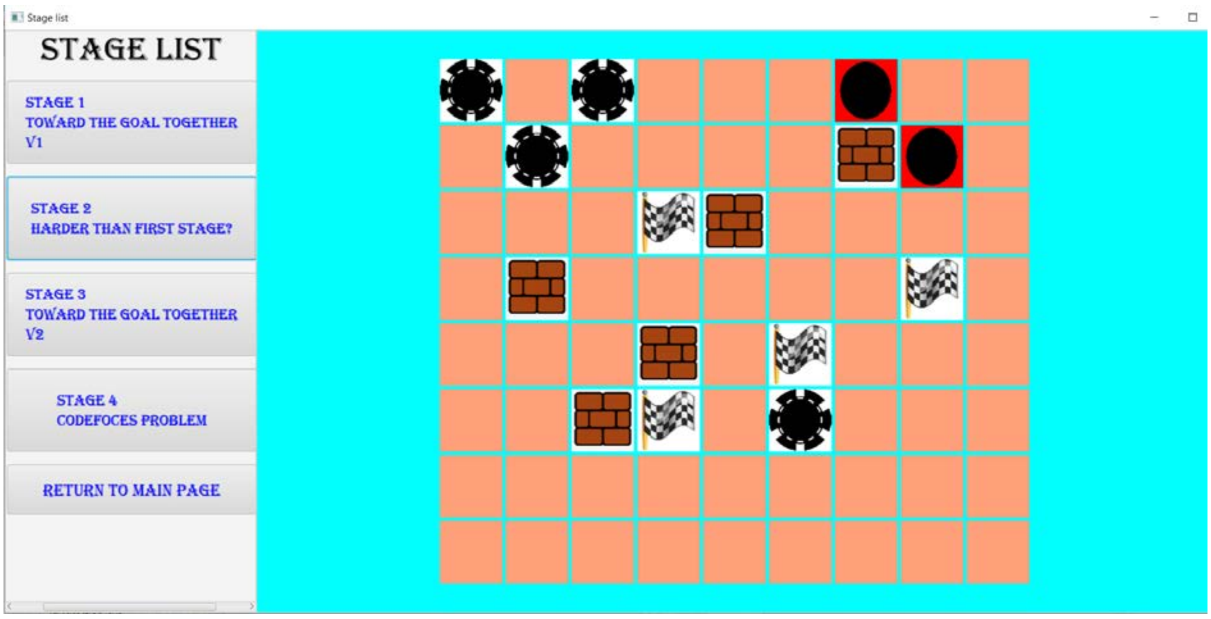
You can click at each phrases.

"Standard maps" will lead you to stage list page.

"Create map" will lead you to simulation page.

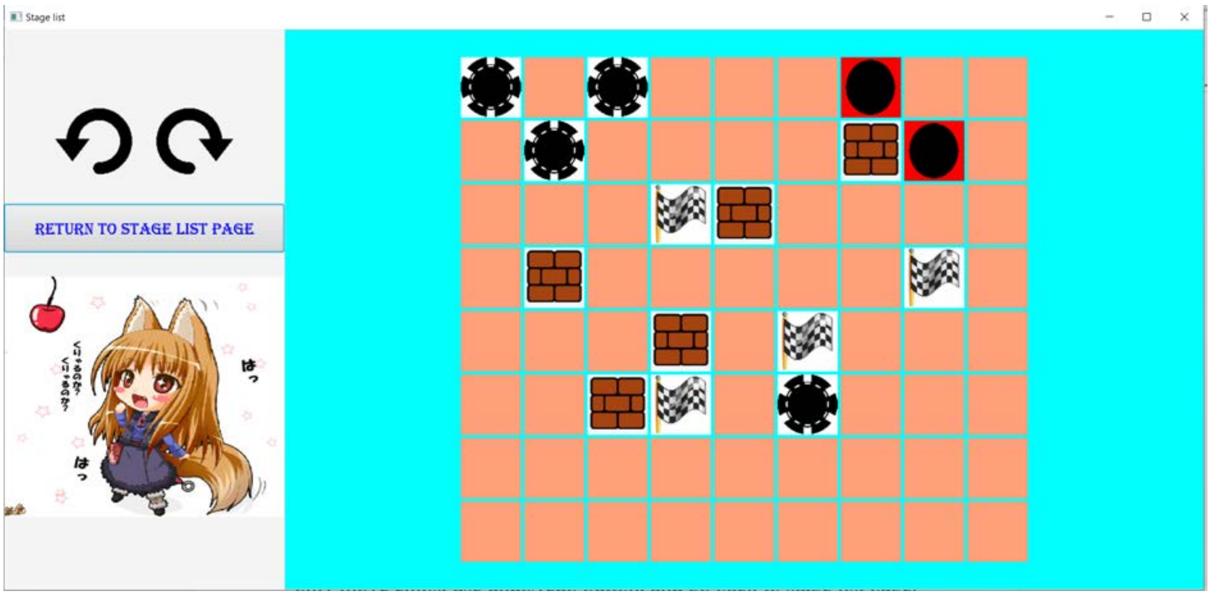
"How to play" will show short instructions of the game.

Stage list scene



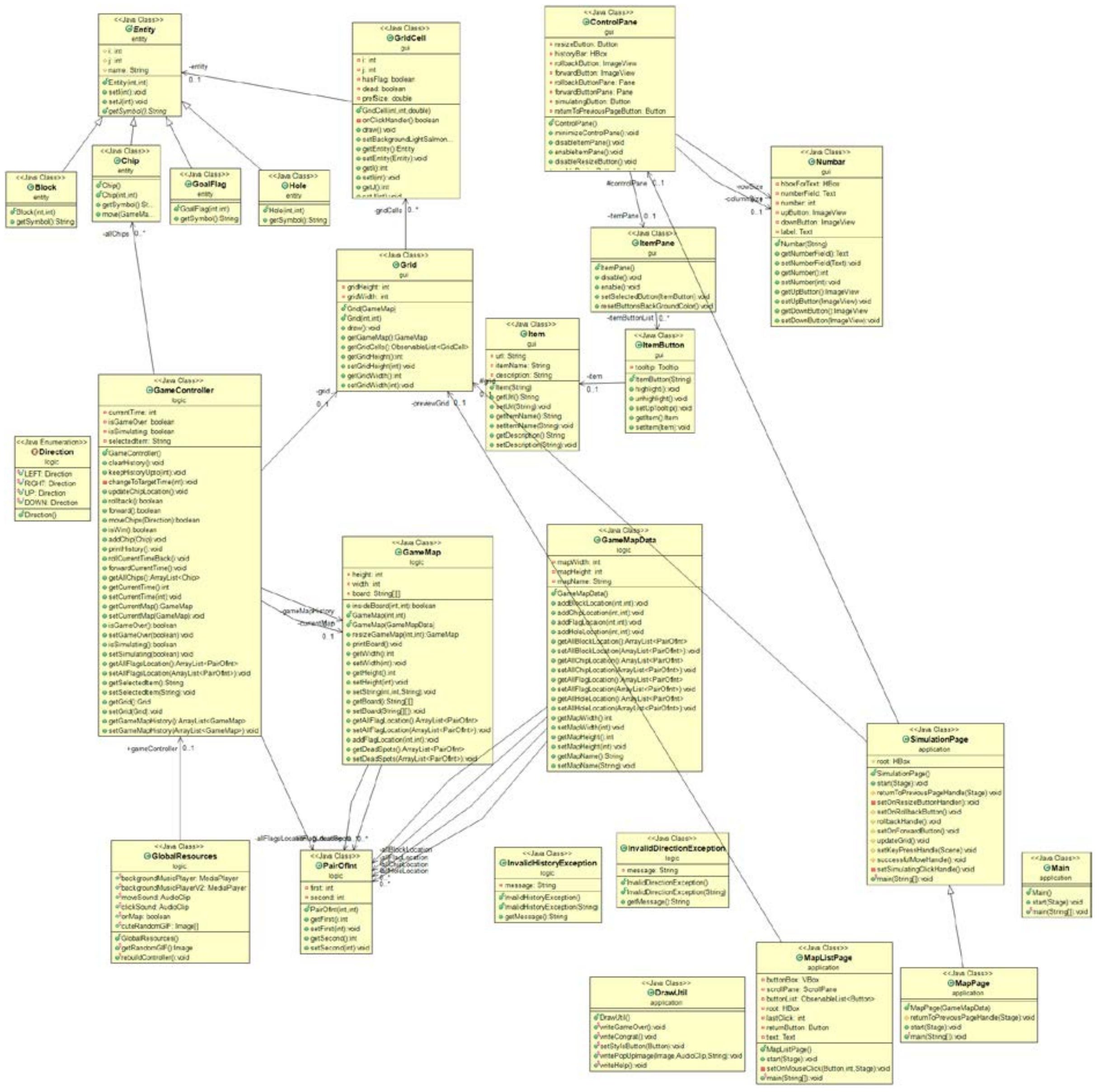
The left side is the button with map name. If you click it, the preview map will be shown in the right side and if you click again, it will direct to that map.

Game map scene



In the map, it will look similar to creating map feature, but you can only move chips, use undo/redo button, and go back to stage list page. Since the left side has too much space, there will be a random GIF shown there.

Class diagram



1. Package entity

1.1 public abstract class Entity

1.1.1 Fields

-int i;	The coordinate by row of this entity.
-int j;	The coordinate by column of this entity
-String name;	Entity’s name

1.1.2 Constructor

+ Entity (int i, int j)	Set row coordinate to i and column coordinate to j

1.1.3 Methods

+public abstract String getSymbol()	Return symbol of entity
+int getI()	Getter for i
+void setI(int i)	Setter for i
+int getJ()	Getter for j
+void setJ(int j)	Setter for j
+String getName()	Getter for name;
+void setName(String name)	Setter for name;

1.2 public class Block extends Entity

1.2.1 Constructor

+ Block (int i, int j)	Set row coordinate to i and column coordinate to j and set name to “Block”

1.2.2 Methods

+public String getSymbol()	Return “B”

1.3 public class GoalFlag extends Entity

1.3.1 Constructor

+ GoalFlag (int i, int j)	Set row coordinate to i and column coordinate to j and set name to “GoalFlag”

1.3.2 Methods

+public String getSymbol()	Return “GF”

1.4 public class Hole extends Entity

1.4.1 Constructor

+ Hole (int i, int j)	Set row coordinate to i and column coordinate to j and set name to “Hole”

1.4.2 Methods

+public String getSymbol()	Return “H”

1.5 public class Chip extends Entity

1.5.1 Constructors

+Chip()	Set row and column coordinate to (0, 0) and set name to “BlackChip”
+ Chip (int i, int j)	Set row and column coordinate to (i, j) and set name to “BlackChip”

1.5.2 Methods

+public String getSymbol()	Return “C”
+public boolean move(GameMap gameMap, Direction direction) throws InvalidDirectionException	Try to move this chip on the gameMap by direction. If the direction is invalid, throw InvalidDirectionException. Return true if the move is successful otherwise return false.

2. Package GUI

2.1 public class Item

2.1.1 Fields

-String url;	url to the image of this item.
-String itemName;	The name of this item.
-String description;	The description of item.

2.1.2 Constructor

+Item (String type)	Set url, itemName, description according to type.

2.1.3 Methods

+Getter, Setter for each field	

2.2 public class ItemButton extends Button

2.2.1 Fields

-Item item;	Item of this button.
-Tooltip tooltip;	Tooltip for showing item's description.

2.2.2 Constructor

+ItemButton(String type)	Set item according to type and setup tooltip.

2.2.3 Methods

+void highlight()	Highlight this button.
+void unhighlight()	Unhighligth this button.
+void setupTooltip();	Set description for tooltip;
+ Getter and setter for each field.	

2.3 public class ItemPane extends GridPane

2.3.1 Fields

-ObservableList<ItemButton> itemButtonList;	ObservableList of ItemButton for this pane.

2.3.2 Constructor

+ItemPane()	Add 6 ItemButtons to itemButtonList and manage click handle for each of them.

2.3.3 Methods

+void disable()	Disable all buttons in pane.
+void enable()	Enable all buttons in pane.
+void resetButtonBackgroundColor()	Unhighlight all buttons in pane.
+void setSelectedButtonHandler(ItemButton selectedItemButton)	Handler for selected button. Set the game controller's selected item to selected button's item.

2.4 public class GridCell extends Pane

2.4.1 Fields

-Entity entity;	Entity of this cell.
-int i;	Row coordinate in Grid.

-int j;	Column coordinate in Grid.
-boolean hasFlag;	Boolean to show that this GridCell has flag or not.
-boolean dead;	Boolean to show whether this cell is dead or not.
-double prefSize;	Size for this GirdCell.

2.4.2 Constructor

+GridCell(int i, int j, double PrefSize)	Set coordinate to (i, j) and set size to prefSize. And setup MouseClick handler for this cell.

2.4.3 Methods

+boolean onClickHandler()	Get selected item from game controller and try to put item into this cell. Return true if item is successfully put, false otherwise.
+void draw()	Set up image and background according to entity, hasFlag, dead.
+ void setBackgroundLightSalmonColor()	Set background of this cell to Color.LIGHTSALMON
+Getter and setter for each field except prefSize.	

2.5 public class Grid extends GridPane

2.5.1 Fields

- ObservableList<GridCell> gridCells;	Observable list for GridCell in this grid.
-int gridHeight;	Number of rows in grid.
-int gridWidth;	Number of columns in grid.

2.5.2 Constructor

+Grid (GameMap gameMap)	Build grid according to data from gameMap.
+Grid(int gridHeight, int gridWidth)	Build empty grid with size of row and column equal to (gridHeight, gridWidth)

2.5.3 Methods

+void draw()	Draw every cell in grid.
+GameMap getGameMap()	Create new GameMap that has same data to this grid and return.

+Getter and setter for each field.	

2.6 public class Numbar extends HBox

The tab for setting number.

2.6.1 Fields

-HBox hboxForText;	HBox for putting text inside and set background and style.
-Text numberField;	Text that show current number.
-int number;	Current number.
-ImageView upButton;	ImageView for increasing number button.
-ImageView downButton;	ImageView for decreasing number button.
-Label label;	Label for Numbar’s name.

2.6.2 Constructor

+Number (String label)	Set label’s text to label and set base value of number to 5.

2.6.3 Methods

+Getter and setter for each field.	

2.7 public class ControlPane extends VBox

2.7.1 Fields

-Numbar rowSize;	Numbar for customizing number of rows.
-Numbar columnSize;	Numbar for customizing number of columns.
-Button resizeButton;	Button for resize grid to current size from rowSize and columnSize.
-ItemPane itemPane;	Normal ItemPane.
-Pane rollbackButton;	Pane for undo function.
-Pane forwardButton;	Pane for redo function.
-HBox historyBar;	HBox for rollbackButton and forwardButton.
-Button simulatingButton;	Button for starting or stopping process of simulation.
-Button returnToPreviousPageButton;	Button for returning to previous page.

2.7.2 Constructor

+ControlPane()	Set up all Panes, Buttons, HBoxes and add them to root.

2.7.3 Methods

+void minimizeControlPane()	Remove everything except returnToPreviousPageButton and historyBar.
+void disableItemPane()	Disable itemPane.
+void enableItemPane()	Enable itemPane.
+void disableResizeButton()	Disable resizeButton.
+void enableResizeButton()	Enable resizeButton.
+Getter and Setter for each field.	

3. Package logic

3.1 public enum direction

3.1.1 enum

{LEFT, RIGHT, UP, DOWN}

3.2 public class EntityComparator

Class that store comparators for sorting chips

3.2.1 Fields

-public static final Comparator<Chip> compareByI	Comparator for sorting chips by row coordinate.
-public static final Comparator<Chip> compareByJ	Comparator for sorting chips by column coordinate.

3.3 pubilc class PairOfInt

3.3.1 Fields

-int first;	First int.
-int second;	Second int.

3.3.2 Constructor

-PairOfInt (int first, int second)	Set first int to first and second int to second.

3.3.3 Methods

+Getter and setter for first and second.	

3.4 public class InvalidDirectionException extends Exception

3.4.1 Field

-String message	Message for exception

3.4.2 Constructor

+InvalidDirectionException()	Set message to “The direction is null. You can’t move”.

3.4.3 Methods

+Getter and setter for message	

3.5 public class InvalidHistoryException extends Exception

Exception class for invalid undo and redo states.

3.5.1 Field

-String message	Message for exception

3.5.2 Constructor

+InvalidHistoryExcention(String message)	Set this.message = message.

3.5.3 Methods

+Getter and setter for message	

3.6 public class GameMapData

Class for storing necessary data for GameMap

3.6.1 Fields

-int mapHeight;	Number of rows.
-int mapWidth;	Number of Columns.
-String mapName;	Name for map.
-ArrayList<PairOfInt> allBlockLocation;	ArrayList that store all block locations.
-ArrayList<PairOfInt> allChipLocation;	ArrayList that store all chip locations.
-ArrayList<PairOfInt> allFlagLocation;	ArrayList that store all flag locations.
-ArrayList<PairOfInt> allHoleLocation;	ArrayList that store all hole locations.

3.6.2 Constructor

+GameMapData()	Initialize every ArrayList.

3.6.3 Methods

+void addBlockLocation(int i, intj)	Add new pair (i, j) to allBlockLocation.
+ void addChipLocation(int i, int j)	Add new pair (i, j) to allChipLocation.
+ void addFlagLocation(int i, int, j)	Add new pair (i, j) to allFlagLocation.
+void addHoleLocation(int i, int j)	Add new pair (i, j) to allHoleLocation.
+Getter and setter for each field.	

3.7 public class GameMap

3.7.1 Fields

-int height;	Number of rows.
-int width;	Number of columns
-String [][] board;	String that represent gameboard. Each entity in the board will represent by symbol.
-ArrayList<PairOfInt> allFlagLocation;	ArrayList for location of goal flag.
-ArrayList<PairOfInt> deadSpots;	ArrayList for location of dead cells.

3.7.2 Constructor

+GameMap (int height, int width)	Create empty GameMap with size equal to (height, row)
+GameMap(GameMapData gameMapData)	Create gameMap according to data from gameMapData

3.7.3 Methods

+GameMap resizeGameMap(int newHeight, int newWidth)	Get new GameMap with size equal to (newHeight, newWidht). The information from current GameMap will be transferred if the location is inside new GameMap.
+void printBoard()	Print board for debugging purpose.
+void setString(int i, int j, String target)	Set String at board[i][j] to target.
+Getter and Setter for each field.	

3.8 public class GameController

3.8.1 Fields

-ArrayList<Chip> allChips	ArrayList that keep all chips.
-ArrayList<PairOfInt> allFlagsLocation	ArrayList for all flag Locations.
-int currentTime = 0;	Denote the steps from moving start.
-GameMap currentMap;	Current GameMap.
-Grid grid;	Current controlled grid.
-boolean isGameOver = false;	Boolean denote whether the game is over or not.
-boolean isSimulating;	Boolean denote whether the game is on active status (moveable) or not.
-String selectedItem;	String denotes the type of selected item. Null is no item is selected.
-ArrayList<GameMap> gameMapHistory;	ArrayList that store all valid history.

3.8.2 Methods

+boolean moveChips (Direction direction)	Try moving all chips according to direction. Return true if at least one chip is moved.
+boolean isWin()	Check if all flags locations are occupied by chips and number of flags > 0.
+void addChip(Chip chip)	Add chip to allChips.
+boolean rollback()	Try to go back to previous states of game if possible. Return true if successfully rollback, false otherwise.
+boolean forward()	Try to go to next states of the game (If rollback before and possible to go.) Return true if successfully forward, false otherwise.
+void forwardCurrentTime()	Set currentTime to currentTime + 1.
+void rollCurrentTimeBack()	Set currentTime to currentTime – 1.
+void changeToTargetTime (int targetTime)	If targetTime < 0 throw InvalidHistoryException(“access negative index”). If targetTime is out of size of valid history throw InvalidHistoryException(“non-allocated memory access”). Otherwise, change game states to the state at targetTime and change currentTime to targetTime.
+void keepHistoryUpTo(int index)	Resize history upto index time.
+void clearHistory()	Clear all history and change currentTime to 0.
+void updateChipLocation()	Update all chip locations.
+void printHistory()	Print all history for debugging purpose.
+Getter and setter for each field.	

3.9 public class GlobalResources

Class for store useful global variable such as current GameController, Images, sounds.

3.9.1 Fields

-public static MediaPlayer backgroundMusicPlayer;	MediaPlayer for main page.
-public static MediaPlayer backgroundMusicPlayerV2	MediaPlayer for stage list page.
-public static AudioClip moveSound;	Sound when Chips move.
-public static AudioClip clickSound;	Sound when button is clicked.
-public static GameController gameController;	Current GameController.
-public static boolean onMap;	Boolean denote whether the current status is on playable map.
-public static Image[] cuteRandomGIF;	Array for .gif image.

3.9.2 Methods

+Image getRandomGif()	Return random .gif image.
+public static void rebuildController()	Set controller to new Controller().

4. Package application

4.1 public class DrawUtil

Class for drawing utility.

4.1.1 Methods

+public static void writeGameover()	Write pop up image for Game Over.
+public static void writeCongrat()	Write pop up image for winning.
+public static void writePopUpImage(Image image, AudioClip audioClip)	Write image as pop up and play audioClip.
+public static void writeHelp()	Pop up help image.
+public static void setStyleButton(Button button)	Set button style.

4.2 public class Main extends Application

Class for main page of game.

4.2.1 Methods

+void start(Stage primaryStage) throws Exception	Play background music. Set up background Image and each button. Set on mouseclick and mousemove for each button.

+public static void main(String [] args)	launch(args)

4.3 public class SimulationPage

Class for simulation page (create map page) of game.

4.3.1 Fields

-protected HBox root;	Root for scene.
-protected ControlPane controlPane;	Current ControlPane of this page.
-protected Grid grid;	Current Grid.

4.3.2 Methods

+protected void setOnRollbackButton()	Set MouseClicked handle for undo button.
+protected void setOnForwardButton()	Set MouseClicked handle for redo button.
+protected void rollbackHandle()	Try rollback game states and update grid.
+protected void updateGrid()	Update grid and gameMap.
+protected void setKeyPressHandle()	Set on keyPressed and keyReleased for moving chips.
+protected void successfulMoveHandle()	Manage updating when chips successfully move.
+private void setSimulatingClickHandle()	Set action when simulating button is clicked. Try switching the game status. If current status is game over, automatically rollback.
+protected void returnToPreviousPageHandle(Stage stage)	Try to go back to previous page.
+ void start(Stage stage) throws Exception	Set on every button’s handle. Managing root and layout.

4.4 public class MapPage extends SimulationPage

Class for standard map play.

4.4.1 Methods

+MapPage (GameMapData)	Try building GameMap and Grid according to GameMapData.
+void returnToPreviousPageHandle()	Try returning to "Stage List" page.
+ void start(Stage stage) throws Exception	Minimize controlPane and add random .gif image to controlPane. Set on each button's handler by parent's method.

4.5 public class MapListPage

Class for stage list page.

4.5.1 Fields

-VBox buttonBox;	VBox for access map buttons.
-ScrollPane scrollpane;	ScrollPane for buttonBox,
-Observable<Button> buttonList;	Observable list for all buttons.
-HBox root	Root for scene.
-int lastClick	Last click button's index. Default 0.
-Button returnButton;	Button for returning to main page.
-Text text;	Text for title.

4.5.2 Methods

+void setOnMouseClick (Button mapButton, int index, Stage stage)	Set on mouseClick for this button for directing to mapPage or preview other maps.
+ void start(Stage primaryStage) throws Exception	Initialize each field and managing layout.

5. Package mapList

5.1 public class MapList

Database class for storing standard maps' information.

5.1.1. Field

-public static ArrayList<GameMapData> mapList;	ArrayList containg all standard maps' information.