Isaac Getto, Pinar Demetci
12 March 2015

Interactive Programming Write-up

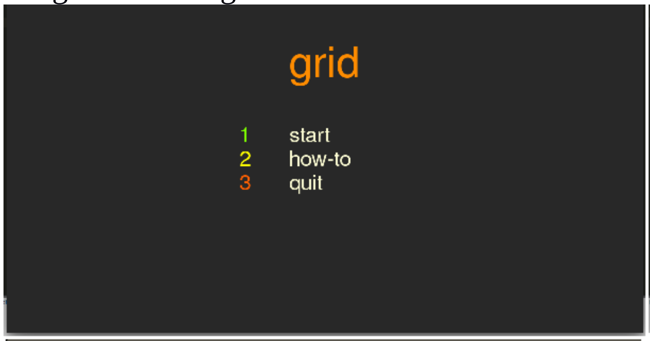
1. Project Overview

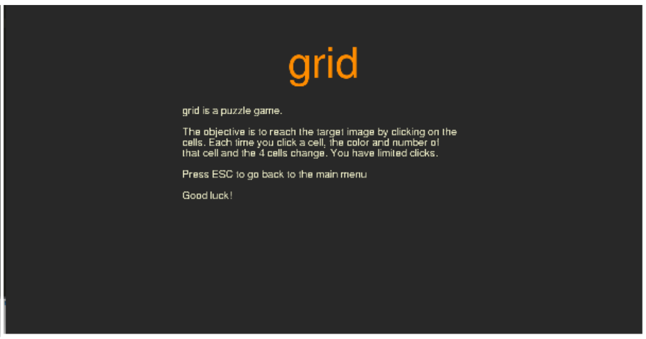
For the interactive programming project, we chose to create a puzzle game using the pygame module. To describe the game a little bit: The puzzle contains a grid of cells. In the beginning, each cell is white. When the user clicks on a cell, the color of that cell and the surrounding 4 cells change. The color change always follows the certain path: white → red → orange → yellow → green → blue → purple. The goal is to match the puzzle to the predefined target image with a limited number of clicks. The game has 4 views: menu, how-to page, the target image, and the puzzle itself. The target image and the puzzle use grid models that define and control each cell's state. Menu and how-to page are controlled by “MenuController” that tracks the keyboard events and the puzzle is mainly controlled by “MouseController”. The design will be detailed in the *'Implementation'* section.

2. Results

The final product is somewhere between the minimum viable product and the stretch goal that we set in the project proposal. Our minimum viable product included only 1 target image, with numbers instead of colors and no undo button. Our stretch goal included colors in the grid, two levels to choose from in the menu (easy and hard) and various target images for these levels. We also planned to have an undo button in our stretch goal, as suggested by Ben Hill.

The final product we created presents different target images each time played. What's missing from the stretch goal is the choice of levels. For the grid, it uses colors instead of numbers since it is more visual. Additionally, we decided to make the screen resizable. Below are some images from the game:



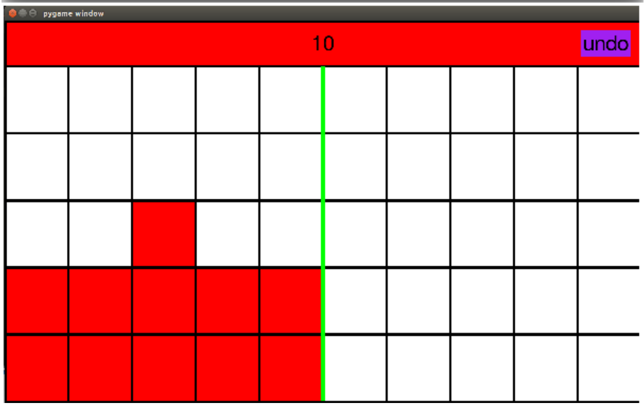

grid

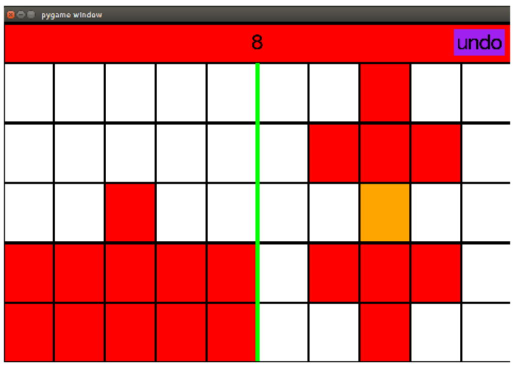
grid is a puzzle game.

The objective is to reach the target image by clicking on the cells. Each time you click a cell, the color and number of that cell and the 4 cells change. You have limited clicks.

Press ESC to go back to the main menu

Good luck!





3. Implementation:

Our project is implemented using a model-view-controller architecture. The UML diagram is at the end of this document. Our models represent the data and the operations that affect it. The views simply display the models. And, the controllers contain the logic to react to user interaction.

The most significant model in our game is the Grid. This class defines a matrix of values and is implemented as a two dimensional list.

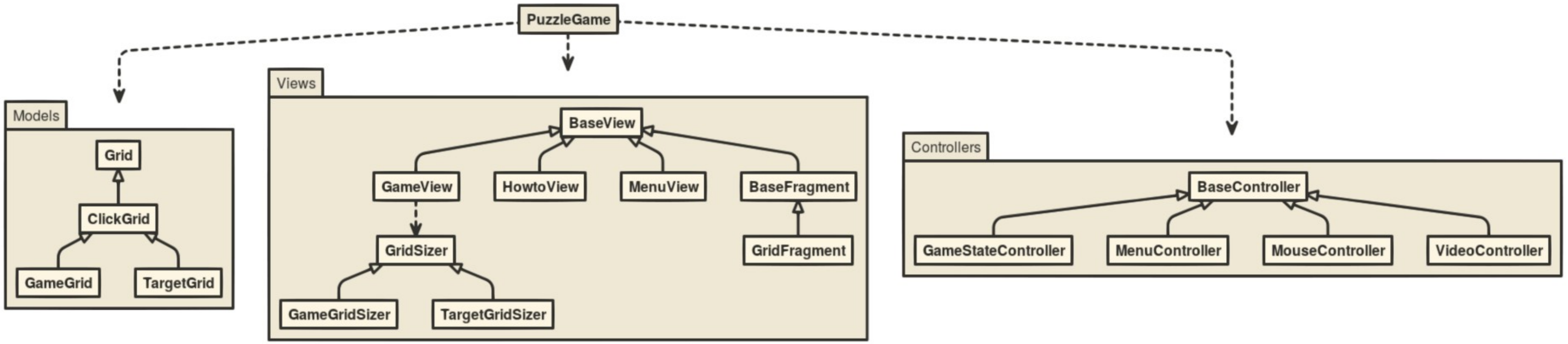
A number of data structures are used throughout the game. In order to implement the undo function, we use a stack. The LIFO protocol allows us to add the player's next move to the stack and then pop it off when they undo their move. Additionally, a map is used to correlate numbers to colors. Within the grid, integers are stored to represent the number state of each cell. A map is used to translate those numbers to colors that are drawn.

In order to handle multiple views, including the menu, how-to, and game views, we were forced to make a design decision. Instead trusting the game make decisions based on the type of the current view, the views make their own decisions. In order to do this, the views implement a common interface. Each view must be able to draw itself as well as handle a mouse click. Part of the trade off of this decision is that the view takes on some additional logical responsibilities. This is not ideal, but doing so allows the other parts of the game to be ignorant of the specific view.

4. Reflection:

Looking at the big picture of the project, we did a good job. We completed the project with the features we specified in the time-frame we agreed upon. The project was finished on time with all of the expected features. We met frequently to work on the game and we were both responsible for our tasks. As well, we were able to produce a functional and fun game.

On the smaller scale, things could be improved. While we both took responsibility for our components, we never conducted a code review or took some time to look through each other’s contributions. Additionally, there could be more testing of each class.


PuzzleGame
Models
Grid
ClickGrid
GameGrid
TargetGrid
Views
BaseView
GameView
HowtoView
MenuView
BaseFragment
GridSizer
GridFragment
GameGridSizer
TargetGridSizer
Controllers
BaseController
GameStateController
MenuController
MouseController
VideoController