ITP368 Project Documentation
Game Title: Destructo-Block
By Jonathan Luu

Application Definition Statement

Destructo-Block is a Java application based off of the existing puzzle game Destructo-Match. It is a single player game with one objective: destroy as many blocks as possible.

Application Rules

The game starts off with a set number of blocks on the screen, and each block is randomly assigned a color. The user must destroy a certain number of these blocks and reach a score of 70 in order to win the game. If they fail to do so, the game is over.

How to destroy blocks

To destroy blocks, the player must click on a block that is adjacent to similarly colored blocks. Clicking on a block once will select all adjacent blocks of that color. If there are no adjacent blocks of that color, nothing will be selected. Clicking again on one of the selected blocks will destroy all currently-selected blocks. Clicking somewhere else on the screen (not on one of the selected blocks) will unselect the currently selected blocks, allowing the user to select a different set of blocks.

Block Types

There will be five block colors generated: red, blue, green, yellow, and gray.

There is also an indestructible block marked by an X that cannot be destroyed. This is limited to three per game.

Block Alignment

Once a block is destroyed, it disappears and the remaining blocks "fall" down to fill in the empty space.

Scoring and Levels

Breaking blocks generates points for the player. Each block is worth one point. There are 84 total blocks generated and 70 blocks must be destroyed in order to progress to the next level.

Initially, the player starts off with only three block types. As the player progresses on to further levels, more types will be added until the maximum of 6 types is reached.

Application Structure

When the game starts, blocks will be randomly generated. There will be a score label at the bottom-left corner as well as a pause button so the user can reset the game. There will also be a score-goal label at the bottom, indicating how many points must be achieved to win.

When the game ends from a game-over, a separate window will appear indicating a win or a lose. The user can then click reset to restart the game or go to the next level if they win.

Menu Screen

When the user opens the application, they start off on the Start-Menu screen. This menu contains three buttons: New Game, Instructions, and Exit.

New Game: Starts a new game. If the player returns to the menu from the main game, they lose all progress and must start a new game.

Instructions: A brief overview of the goal and instructions of the game. Displayed through a new dialog pop-up that can be closed by pressing the “Close Instructions” button.

Exit: Completely exits the game and closes the window

Sound

When the user opens the application, the soundtrack will be played immediately and will loop indefinitely.

Additionally, when the user destroys a set of blocks, a “crackling” sound will be played.

All sounds are controlled by the MusicSingleton controller.

Pause Menu

After the user starts a new game, a pause button will be displayed on the top-right corner of the application. If the user presses this button, a dialog pop-up will appear with three buttons: Continue, New Game, and Return to Menu.

Continue: Closes the dialog to resume gameplay

New Game: Resets the game completely, starting the player at level 1 with a score of 0. Block-type variety is reset to its initial value of three as well.

Return to Menu: Resets the game and sends the user back to the main menu screen.

Entities

1. Block

Description: Basic block type. Has a color and can be destroyed by if adjacent block with same color exists.

```
class Block{
        private Pair<int,int> mLocation;
        private Image mImage;
        private bool mIsSelected;
        private bool mIsDestroyed;

        Block();
        GetLocation();
        GetIsSelected();
        GetIsDestroyed();

        SetLocation();
        SetIsSelected();
        SetIsDestroyed();
        SetImage();

        private void addImageListener();
        private void moveBlock();
        private void fadeBlock();
}
```

2. Blue Block

Description: Basic block type. Has a color blue and can be destroyed by if adjacent block with same color exists.

```
class BlueBlock{
        private String mColor;
        BlueBlock();
}
```

3. Green Block

Description: Basic block type. Has a color green and can be destroyed by if adjacent block with same color exists. See BlueBlock for class definition.

4. Red Block

Description: Basic block type. Has a color red and can be destroyed by if adjacent block with same color exists. See BlueBlock for class definition.

5. Yellow Block

Description: Basic block type. Has a color yellow and can be destroyed by if adjacent block with same color exists. See BlueBlock for class definition.

6. Gray Block

Description: Basic block type. Has a color gray and can be destroyed by if adjacent block with same color exists. See BlueBlock for class definition.

7. Indestructable Block

Description: Special block type. Cannot be destroyed. Only three can be created per game

```
class IndestructableBlock extends Block{
        IndestructableBlock();
        SetColorNull();
}
```

8. Background

Description: Background image for the game.

```
class Background{
        private Image mBackgroundImage;

        Background();
        ClearImage();
        SetImage();
}
```

9. Game Over Panel

Description: Panel with one button that is displayed when the game ends. Allows the user to reset the game

```
class GOPanel{
        private Button returnButton, submitButton;
        GOPanel();

        Reset();
}
```

10. Pause Panel

Description: Panel with various buttons that is displayed when the game is paused by the user. Can either continue or reset the level.

```
class PausePanel{
        Continue();
        Reset();
```

}

11. Pause Button

Description: Button that invokes the Pause Panel. Only displayed during the game, not on the main menu.

```
class PauseButton{
        private Image pauseImage;
        PauseButton();

        OpenPanel();
}
```

Controllers

1. Block_Singleton

Description: Singleton class that maintains the game state. Will contain an arraylist of all blocks that are currently in the game and can be accessed by other classes to update the display. Will also contain the current totalScore and current level.

```
class Block_Singleton(){
        private static Block_Singleton sBlock;
        private ArrayList<Block> allBlocks;
        private Integer totalScore, currentLevel;

        private Block_Singleton();
        public static Block_Singleton get();

        public getAllBlocks();
        public getTotalScore();
        public addNewBlock();
        public removeBlock();
        public getBlockSize();
}
```

2. Game Controller

Description: Singleton class that maintains the game state. Constantly checks if there are any remaining moves in the game. If not, it creates the Game Over Panel and determines whether or not the player has won or lost.

```
Class Game_Controller(){
        private static Game_Controller gameController;
        private Stage passedStage;

        Game_Controller();
        get();

        check();
        setStage();
        getStage();
}
```

3. Music_Singleton

Description: Singleton class that maintains the sound of the game. Can be called from anywhere to invoke a sound effect or play the background music.

Views

1. Draw_Panel

Description: The main game panel. This is where the application launches (public static void main()). Does some basic setup of the scene and creates the various elements of the game.

```
public class DrawPanel extends Application{
	public void start(Stage stage){}

	public static void main(String [] args){
		launch(args);
	}
}
```

2. Score_Panel

Description: The bottom part of the main game panel. Contains the current score and score goal. In order to progress to the next level, the player must reach the score goal. Also contains power-ups for the current game and the current level.

```
public class ScorePanel{
	public ScorePanel(BorderPane bp);

	private void DrawBar(); //Draw the grey panel at the bottom
	private void DrawText(); //Draw text for the bottom bar
	private void DrawPowerUps(); //Draw power-up boxes
}
```

3. Block Panel

Description: The center part of the main game panel. Contains the blocks currently in the game. It creates the blocks and also displays them.

```
public class BlockPanel{
	public BlockPanel(BorderPane bp);

	public static void Redraw(); //Allow blocks to call this function whenever they die
	public static void DrawColumn(); //Draw a single column of blocks
	private void CreateSingleBlock(); //Creates a random colored block
	private static int randInt(int min, int max); //Helper function to create a random #
}
```

3. Start Menu Panel

Description: The initial start menu of the game. Contains the three buttons that the user can press to start the game: new game, instructions, and exit

```
public class StartPanel{
	public StartPanel(BorderPane bp);

	public void addButtons();
}
```