ITP368 Project Outline
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 in order to progress to the next level. If they fail to do so, it is game 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.

### *Block Types*

Only three block colors will be generated initially: red, blue, and green. As the user progresses on to further levels, more colors may be added to increase the difficulty.

There will also be 'special' blocks that the user can tap. For example, there will be a 'bomb' block that destroy several blocks around the bomb block. Another example would be a color- elimination block, which destroys all of that color block in that level. These blocks are placed randomly throughout each level.

### *Block Alignment*

Once a block is destroyed, it disappears and the remaining blocks "fall" down to fill in the empty space. If an entire column is destroyed, blocks to the right of that column will "slide" over to the left to fill in the gap.

### *Scoring*

Breaking blocks generates points for the player. Breaking more than two blocks with a single selection creates a multiplier that scales depending on the number of blocks destroyed. This encourages the player to destroy as many blocks as possible with each selection, rather than clicking mindlessly.

# Application Outline

The user initially starts out on the main menu screen. There will be four options displayed:

1. New Game
    a. Starts the player at level one
2. Load Game
    a. Starts the player where they left off. If the player quits their current game by clicking on the cog icon/pressing escape and then selecting “Save and Quit”, the game will save the current game’s state and resume it if the player selects the “Load Game” option from the main menu.
3. Scores
    a. Display a menu of top local scores by the user. A potential idea is to have an online score section where scores are sent to a MySQL server and then fetched accordingly to display to all users.
4. Settings
    a. Displays various options for the user to adjust.
        i. Name: Set your name. Mainly used for the Scoreboard.
        ii. SFX/BGM sliders: Allow the user to adjust sound effects/background music levels
        iii. Mute checkbox: Disregard SFX/BGM sliders and turn all sound off
        iv. Reset: Erases all local scores
        v. Instructions: Opens an in-game popup that displays the rules (possibly with images for clarity)

When the game starts, blocks will be randomly generated (for a new game) or re-generated from a previously saved game. There will be a score label at the top-left corner as well as a pause button (Cog icon/pause icon) so the user can quit, adjust settings, start a new game, or save the current game.

When the game ends from a game-over, a separate window will display the user’s final score and rank. There will be two buttons at the bottom: Return to Menu and Submit Score. If the user presses Return to Menu, it brings them back to the main menu. If they hit submit score, it will submit the score to the MySQL server for other users to see for the online leaderboard.

# 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 mBlockImage;
        private bool mIsSelected;
        private bool mIsDestroyed;

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

        SetLocation();
        SetIsSelected();
        SetIsDestroyed();
        SetImage();
}
```

## 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 coloryellow  and can be destroyed by if adjacent block with same color exists. See BlueBlock for class definition.

## 6. Bomb Block

Description: Special block type. Has a bomb image instead of a color. Destroys all adjacent blocks around it (including diagonal ones) if they exist.

```
class BombBlock extends Block{
        BombBlock();
        SetColorNull();
        DestroyAdjacentBlocks();
}
```

## 7. Star Block

Description: Special block type. Has a star image as well as a color. If destroyed, all other blocks in the level with that color are also destroyed.

```
class StarBlock extends Block{
        StarBlock();
        DestroyAllColorBlocks();
}
```

## 8. Timer Block

Description: Special block type. Has a timer as well as a color. Needs to be destroyed within the time limit (15 seconds) or else it will turn into a solid block that cannot be destroyed.

```
class TimerBlock extends Block{
        private int mCurrTime;

        TimerBlock();
        CountDown();
        TransformToSolidBlock();
}
```

## 9. Indestructable Block

Description: Special block type. Cannot be destroyed. Is created by a timer-block that has reached a count of 0.

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

## 10. Scoreboard

Description: Panel displayed when user selected “Score” on the main menu. Contains list of local scores.

```
class Scoreboard{
```

```
        private Vector<Pair<String, int>> mAllScores;

        Scoreboard();
        DisplayScores();
        ResetScores();
}
```

## 11. Background

Description: Background image for the game. Can change from level to level.

```
class Background{
        private Image mBackgroundImage;

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

## 12. Game Over Panel

Description: Panel with two buttons that is displayed when the game ends. Return to menu or Submit score buttons.

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

        ReturnToMenu();
        SubmitScore();
}
```

## 13. Pause Panel

Description: Panel with various buttons that is displayed when the game is paused by the user. Quit, adjust settings, start a new game, continue, or save the current game buttons.

```
class PausePanel{
        private Button settingsButton, quitButton, newButton, continueButton, saveButton;
        PausePanel();

        OpenSettings();
        ReturnToMenu();
        RestartGame();
        ContinueGame();
        SaveGame();
}
```

14. 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();
}
```