Recycler Robbie Front End Team

CSCI 598SE

Fall 2012

# Program Documentation

**To Run Game**

Run main() from GameLauncher

**Interaction With Kinect/Mouse**

Inside the main function in GameLauncher() use this code:

Run main() from GameLauncher

*To run with mouse:*

ModalMouseMotionInputDriver mouse = new ModalMouseMotionInputDriver();

gameLauncher.getGameManager().installInputDriver(mouse);

*To run with Kinect:*

OpenNIHandTrackerInputDriver kinect = new OpenNIHandTrackerInputDriver();

gameManager.installInputDriver(kinect);

The Kinect/mouse input is handled in PlayerHand.getNextPosition() method in order to update the hand’s position.

The AI input is generated by the AI and is similarly retrieved in ComputerHand.getNextPosition().

**Interaction With Splash Screen**

The interaction of the splash screen mostly takes place in three locations: GameLauncher, GameLogic, and, from the splash screen’s SplashScreenLauncher.

*Starting the game from the SplashScreen*

The game is initially started from the SplashScreenLauncher which sets up a GameLauncher object. This object is what starts the game for the player(s).

*Adding high scores*

When a player has lost the game, a SavePlayer object calls the submitPlayerScore function inside GameLogic’s handleScore() method. This is only called if the player is actually a human player and not the computer.

*Ending the game*

Once the player score has been submitted to the splash screen, isPlaying is set to false in GameLogic.The GameLauncher is continually checking if the both players have lost the game (or if the one player playing against the AI has lost). Once either condition has been met, the GameLauncher returns a null value. This null value will be checked by the splash screen and turns functionality back over to the splash screen.

**Item Generation and Motion**

Items are created independently of the rest of the game using an ItemFactory inside the main updateThis method of GameLogic. After generation, items should be given to an ItemMover using ItemMover.takeControlOfMoveable (or Moveables for a list of items).

The game has no concept of inertia. Moveable items are “tugged” by ItemMover classes during each update loop. ItemMovers have paths they follow and are responsible to calculate new positions for the items they control. ItemMovers are expected to release control of items at the end of the path with releaseControlOfMovablesAtEndOfPath and the GameLogic should either remove them or pass them to a new ItemMover. ItemMovers must additionally released items that are being collided with using releaseTouchableItemsAtPoint. All of these methods are handled inside ItemMover.java and are marked as final - not intended to be extended. What is extendable is moveItems(). Subclasses should use this to calculate new positions of their items and perform any special behavior. For example, the ConveyorBelt also modifies items’ touchability. They do \*not\* handle being displayed on the screen; after ItemMovers release items, other program behavior is expected to remove them from the screen.

**Computer Player**

If a one player game is selected, they play against the AI. GameLogic calls the ComputerPlayer’s updateAI function determines the behavior of the computer player. The updateAI function has three main functionalities: setting the hand to the correct side of the conveyor, following a touchable item up the conveyor, and striking an item. The hand must occupy the same area as the item in order for a collision to be detected.

The AI behavior is determined through the use of random numbers. There is a chance that it will set the hand to the incorrect side or that the result of a collision will result in a strike. These features help the AI emulate a human player. The AI doesn’t learn as gameplay progresses. This is because the motion of the conveyor provides an advantage to the AI later in the game.

## graphics.zip

**/items** - these are the various sprites that are used for the sortable items. These are 100x100 and PNG to utilize a smooth, anti-aliased transparency.

**/layout** - these are the interface files, including the various bin states and frames. These are 1920x1080 to take up an entire 1080p screen. This allows for easy placement from within the game (simply place at 0,0) to where every item lines up as expected.

**layout.c4d** - this is a Maxon Cinema4D R10 project file that was used to create all the layout renders.

**layout.ai** - this is an Adobe Illustrator file that was used to create all the layout sprites. It takes the renders from layout.c4d (once isolated) and lays them out in the correct location. The logo is also contained in this file.

**items.ai** - this is an Adobe Illustrator file that was used to create all the item sprites.