TacoTime Game Engine Overview

Written I. Yulaev 2012-07-22
Contact ivan@yulaev.com for questions

1 Overview

This document outlines the TacoTime game. A description of the game mechanics as well as a high-level overview of the code organization is provided.

2 Game Mechanics

The basic premise of the TacoTime game is that the player acts as a service employee employed in a café/restaurant/food truck. She operates some machinery (like a coffee machine or an oven), putting together various food/food-like items. A queue of customers forms in front of the establishment, and the customers voice their requests. The player must guide their service employee actor to prepare the necessary foodstuffs. Fulfilling customer requests provides some sort of reward (money and heart-points); the faster the requests are filled the more points the employee obtains.

3 Code Structure

3.1 Entry Point and Game Launch

The top-level file of the game is TacoTimeActivity.java. However, this file does very little except for instantiating MainGamePanel. This class creates the Canvas (the 2D area where the game will be drawn), instantiates the game elements, and launches all of the threads.

3.2 Thread and Game Control

Several threads operate simultaneously for TacoTime. They are

1. ViewThread – this thread is responsible for re-drawing all of the game items. Also it has awareness of the game grid (relative position of all items) so it can detect physical item interaction.
2. TimerThread – Does nothing currently. Intended for use to track game time and allow other threads and Objects to deal with time-dependent events.
3. InputThread – receives user input and sends messages to GameItems describing what input occurred
4. GameLogicThread – receives messages from ViewThread; determines the outcome of interactions between the Actor (CoffeeGirl) and GameItems and in general drives game logic.

Also, a MessageRouter exists to allow different objects to pass messages to each other. Message types include

1. InputThread.MESSAGE_HANDLE_ONTAP: passed from MainGamePanel to InputThread with details on what kind of user interaction has occurred.
2. GameLogicThread.MESSAGE_INTERACTION_EVENT: passed from ViewThread to GameLogicThread indicating that CoffeeGirl entered a GameItem's "sensitivity area" AND that the GameItem has a pending interaction queued, which was successfully consumed.

3.3 Game Objects & State

There are two classes of game object – the CoffeeGirl, which is the in-game character (Actor) represented by the player, and GameItems, which are non-player interactive objects. Both types of game objects may (but are not required to) have state.

CoffeeGirl's state transition table is determined by GameLogicThread.coffeeGirlNextState(). This method takes the previous state of CoffeeGirl, the name (ID) of the GameItem that she interacted with, and the state of that GameItem. It then determines the next state of CoffeeGirl. Each CoffeeGirl state has an associated bitmap (later, sprite).

GameItems track state internally. A GameItem (logically) operates as a state machine. State may change due to two reasons: sufficient time has passed and/or interaction with CoffeeGirl has occurred. The state gets then updated. States get updated when GameItem.onUpdate() or Gameitem.onInteraction() by ViewThread and GameLogicThread respectively; GameItems will check whether sufficient time has passed and whether interaction has occurred and determine whether or not to change states. Each GameItem state has an associated bitmap that signals to the user the state of this GameItem.

The State object describes a state. It identifies a State with a name, a bitmap to draw the GameItem with, and the criteria for advancing to the next state(interaction or time sensitivity, delay time).

3.4 Adding Game Objects

To add a new FoodItem, such as coffee or cupcake:

1. Create an icon for it, add it to the relevant sprites so they can be shown carrying/holding it
2. Create a new type representing that food item (extends GameFoodItem); make sure to run GameLogicThread.addNewFoodItem
3. Update MainGamePanel
4. Update GameLogicThread.coffeeGirlNextState() so that CoffeeGirl can get the relevant food item when necessary
5. Add a new state to CoffeeGirl, both in the Class definition (final int) and in the constructor

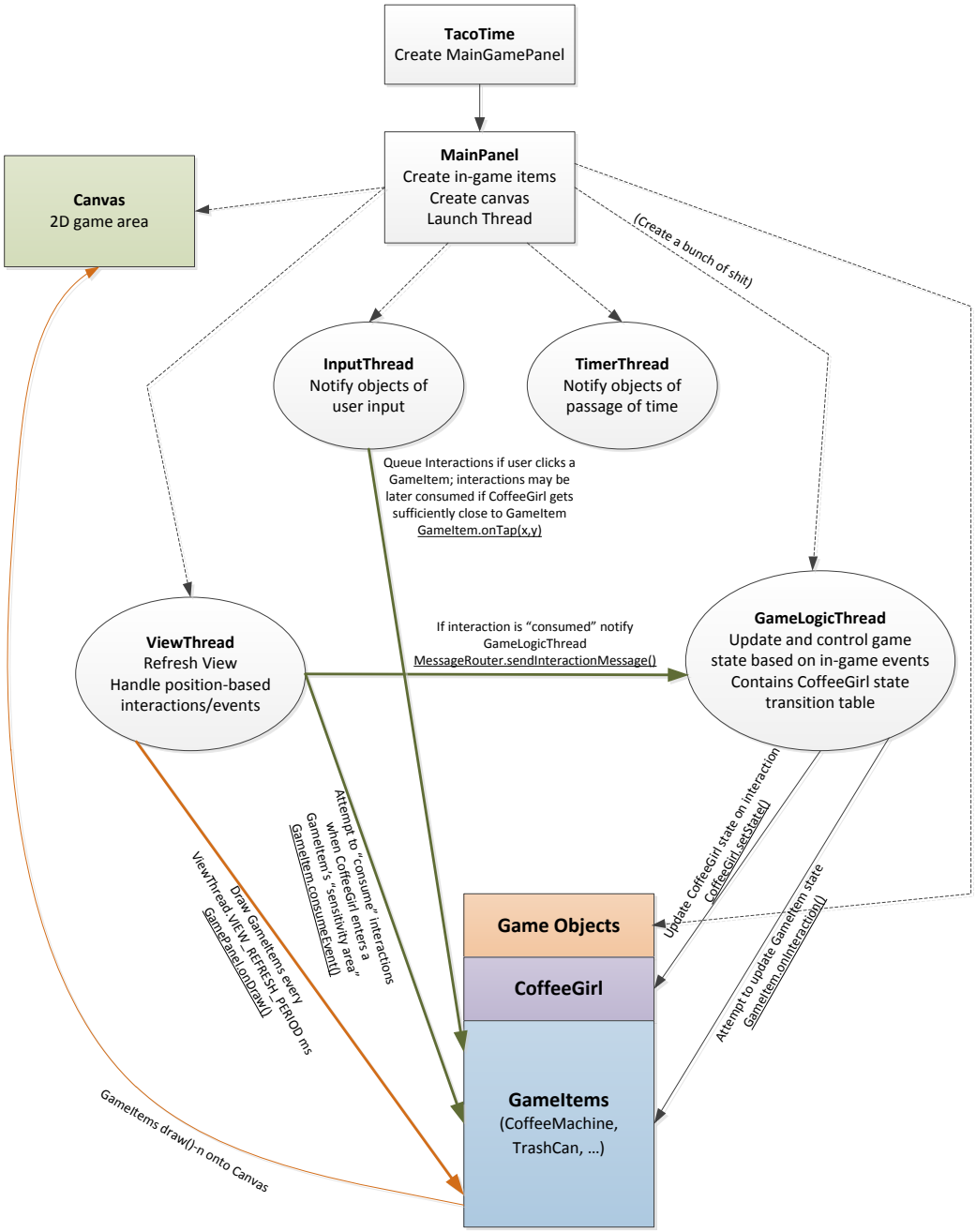
Adding a new game item, such as a Coffee Machine or a Blender:

1. Create an icon for it
2. Create a new type representing that game item (extends GameItem)
3. Update MainGamePanel to initialize and place the game item, and add it to all of the Threads

4. Update GameLogicThread.coffeeGirlNextState() to show how interactions with the gameItem should work

3.5 Diagram of Project Structure

The below diagram illustrates how these threads and game objects fit together.



4 References

Below are some useful references for Android game development.

Writing real-time games
Android Game Development - I lifted some of the code from the beginning parts of this tutorial, although it's mostly been stripped except for the boilerplate.
Android hardware acceleration
Android using nine-patch