Weekly log

Michal Staniaszek

Last updated March 10, 2011

1 [2011-02-01 Tue]-[2011-02-07 Mon]

1.1 [2011-02-01 Tue]

- Worked with Jere and David to discuss the class structure. Also created the top level abstract classes that we will use to create the game.
- Spent 30 mins trying to work out how JFrames work and how they get components added to them. Seems strange. Also thought about how best to do GUI elements. Would be good to have frames and panels in the same package, just because it makes sense. We could use multiple panels for the game, menus and so on, and add them to the main frame or sub objects when appropriate.
- Wrote a basic class for the main frame and tried to work out how the hell to test it, but didn't come up with anything.
- Started writing a panel on which to draw the game screen. This will need to have some sort of way of getting all of the objects that we need to draw from the game object. The game object should have references to all objects that are part of the game. Perhaps two arrays, ships and projectiles, will be good, but not sure if that's a good idea. Seems ok though. Another idea is to have everything that is part of the game as a subclass of some top level object (gameObject?), so that everything can be stored in one array. Again, not sure. Some experimentation will be necessary.
- Thought about how to do the mouse and key listeners. Most likely the best way is to have the control schemes subclassing their respective listeners so that we can pass those to the addlistener methods, which will supposedly make things easier. It might make it difficult to pass

around parameters, but i guess since we're subclassing the listeners it doesn't really matter too much, but that's still to be seen. Otherwise, we just implement the mouselistener and keylistener in the main frame class. The problem of referencing can be solved by having references to the listeners in the frame, and having them chuck back values when something happens. This brings up another problem, though - how do we know when things have changed - we need another listener, which is silly. Might be best to do things the simple way first, and then try to abstract it later. Have no real idea how to solve this at the moment.

- THERE IS WAY TOO MUCH THAT CAN GO WRONG/BE HARD TO DO/BE ANNOYING! AAAAAAAGH!

1.2 [2011-02-03 Thu]

- Started work on the game class. This must integrate data from all over the place and then redistribute it to whichever class wants it. Will have to have a lot of methods to do updates on arrays, collision detection. Not really sure of the complete layout yet, since a lot of the rest of the code is still incomplete.

1.3 [2011-02-04 Fri]

- Thought about ways to solve the problem of fire rate of the vehicle. It shouldn't be too difficult to change the way that the player fires. There is an easy (hacky) way, where you use a count variable to check how many times you've entered the paintcomponent, and then only fire the weapon, say, every 20th time you enter.

- Mouse sensitivity may also be an issue. We may need some way to change the speed of the mouse's motion. A simple solution which looks promising is to take a percentage of the distance the mouse has moved and then modify player location to that value rather than the actual mouse location. This does mean that the mouse would have to be locked to the window, and also the player would run out of space, since you'd stop receiving new mouse input after a while because you reach the edge of the screen. You'd only be able to use a percentage of the window (the percentage of the mouse movement that you were using.)

2 [2011-02-07 Mon]-[2011-02-16 Wed]

2.1 [2011-02-11 Fri]

- Worked on pruning methods for the game class to remove units from the arrays when they are no longer needed. We should probably add some sort of flag to units if we want them to be able to move off screen and reappear somewhere else. This should require some sort of interface with the panel class, otherwise the game will have no knowledge of frame bounds. Could be done with an array pruning method in the game class which takes a width and height parameter, and removes anything from outside that box.

- 3 hours

2.2 [2011-02-12 Sat]

- Wrote JUnit tests for the Projectile class, and for the Game class methods that I wrote. Also wrote classes for use in testing - they take simplified parameters, just locations, so that things to do with location are easier to test rather than having to pass all the necessary parts of the constructor.

- 2 hours

2.3 [2011-02-15 Tue]

- Tried to work out how to do projectiles with Dave.

- Fixed having to create a new object each time you wanted to move the player around.

- Worked on drawing simple projectiles, by implementing a domove method to be called in the paintcomponent class to update the projectile's location, and then draw the updated thing.

- 3 hours

2.4 [2011-02-16 Wed]

- Worked on improving the current JUnit tests that have been written by adding more test cases to the tests and adding more comments.

3 [2011-02-17 Thu]-[2011-02-24 Thu]

3.1 [2011-02-17 Thu]

- Discussed implementing a game logic loop for the game to make it easier for us to control exactly what is going on. This will require a little bit of restructuring of the game panel class - one method to control logic, and another to draw the results of the logic execution. The nice thing about this loop is that we can then pause it if we want to go into a menu.

- Discussed spawns - each spawn needs to have a location, and an enemy type that it spawns, at the very minimum. Later implementations could have multiple enemy types to choose from randomly, and have a timer, or a method of listening to how many time thes main game timer has ticked, and spawning enemies based on that.

- Added a spawns array to the game class to allow for cleaner interaction with the gamepanel class. Each spawn should have some method which will notify that it has spawned objects so that the game can add the objects it has created to its arrays.

- Working on collision detection this week. Most likely going to go for very simple stuff for now. Just check whether the shape of the projectiles overlaps anything in the units, and also check that the projectile was propagated by the correct thing to destroy that object.

3.2 [2011-02-20 Sun]

- Discussed the method of spawning objects with Dan. Do we spawn randomly? Are the spawns fixed? How do we regulate the spawning?

- Spawns should probably be fixed, or we at least should be able to specify the starting location.

- Spawning regulation is unsolved. We could count the number of game loops passed in each spawn, and then spawn enemies based on how many loops we've been through. E.g. spawn an enemy every 100 game loops (approx 3 seconds). Alternatively, a global timer which each spawn class looks at and spawns things every so often.

- How to decide when to spawn stuff?

- Game checks all spawns each game loop for a flag that indicates that they want to spawn, and then if they do, reads off how many units to spawn and spawns them?
- Same flag idea, but game calls a method on the spawn which creates a unit to give to the game class? - probably better.

3.3 [2011-02-22 Tue]

- Discussed the game not drawing projectiles with dave. Seems like the game doesn't want to draw any lines whatsoever, even when inside the paintcomponent directly.
- Started work on the collision detection. It naively goes through all the arrays and creates hitboxes of rectangles of 5x5 around the location of the unit, and checks whether these intersect.
- Finished collision detection method. Does work to some extent, but probably will have many bugs, since not checking whether a projectile is actually supposed to destroy and object. This can probably be solved by checking the isEnemy thing, but not really sure yet.
- Tried dividing the arrays up a lot more, into seperate ones for friendly projectiles and enemies, and friendly units and enemy units. This made a massive mess, so I reverted. This may be a viable method of doing stuff, but at the moment everything else is just as broken, so we should fix that first.
- Projectiles seem to get drawn properly now, although there are still some problems to sort out.

3.4 [2011-02-23 Wed]

- Wrote a fix for the collision detection problems, so now the player cannot destroy itself, and enemies cannot destroy themselves either.
- Wrote methods in the path class and elsewhere to allow for the movement of enemies. The paths take a location and then provide the next point at which that object should be along the path.

3.5 [2011-02-24 Thu]

- Wrote a new path class for simple straight paths.
- Decided that I don't like the current implementation of the directionality of movement. There should be some sort of better way to do this. Maybe there should be a simple path class, which extends path, and has these directions in it. This would make sense, since more complex paths may not have directionality? Thinking again, they probably will, so probably the main path class should have all the possible directions, and then we can use those later to define what the paths return.

4 [2011-02-25 Fri]-[2011-03-03 Thu]

4.1 [2011-03-01 Tue]

- Started work on restructuring the game class a little bit. Split the units array into an array for enemies and players. This should allow for easier handling of some stuff, I hope.
- Started writing a method to collide player and enemy ships. Must be careful not to collide a ship with itself.
- Added a giveDamage method to the Unit class. This is used to damage a unit. No contingencies in here, though. If the health drops below zero, nothing happens yet. Also added a gethealth method, since there wasn't one before, for some reason.
- Fixed the problem where the player is removed from the screen after moving off it - side effect of splitting the unit arrays.
- Changed the time at which enemies spawn - makes testing quicker.

5 [2011-02-04 Fri]-[2011-03-10 Thu]

5.1 [2011-03-08 Tue]

- Tried to start working on the menu stuff, but got distracted by the game over message box not working. Trying to implement some way of throwing stuff to the top level frame to allow the box to work properly, but this seems to be impossible due to the way that the gamepanel currently works.

  - Seems like this was some random incorrect implementation. Fixed by throwing exceptions from the game class up to the panel, and then using the panel to display the lose condition. Don't know why it didn't work before.

- Exceptions are going to be needed if we want stuff to run smoothly, otherwise everywhere is going to have a ton of code that doesn't really fit there, so I've made a few of these and started implementing some stuff to do with that.

- Changed some stuff in the gamepanel to allow throwing of exceptions up to the baseframe. This is needed for switching to menus. The initialisation method has been changed from private to public to allow the baseframe to catch exceptions thrown by it.

- Added a start dialogue to the initialise method of the game panel so that the game doesn't start running straight away.

- Managed to find an insanely hacky way to get exceptions thrown to the base frame. When an exception is thrown, it is put into a variable in the gamepanel class. The frame constantly loops through a method which, if the exception is not null, throws the exception to the frame to handle it.

  - Pretty difficult to implement, or at least it seems so. The exception doesn't seem to want to go up to the frame. Might use a boolean instead.
  - Went with the boolean approach. Seems unnecessary to do it the other way if it makes it more complicated.

- Worked on the panel switching methods in the frame. Used a cardlayout and some enums to specify which panel is being switched to or from, and then deal with the necessary stuff each time the panel switches. The game must be paused when switching out, and unpaused when switching in. The menu probably will require something done, but nothing right now.

- Added a pausing mechanism by allowing the user to stop the game logic and rendering from running. Not worked out a way to get it to restart again yet though.

- Fun time with booleans. Implemented the restart with a boolean that tracks whether the game is paused, and then checks if the pause button has been pressed again, and unpauses if it has.