Team RPGeeze

COP4331

2/20/09

Iteration 1 Post Mortem

As a team we discussed what went well with the first iteration, what didn’t go so well and what we would do differently or improve for next iteration. During the meeting this is what we agreed upon and what we came up with.

One aspect of the iteration that went very well was having a platform built early and on a subversion server so that all the team members could access a working barebones version of the model-view-controller in order to add functionality. This allowed all members to test their code as soon as it was written and reduced dependencies on the completion of code from other teammates. An early design provided scalability. In having a small working system early it was easier to make it bigger and more complex. Another aspect of this iteration that went very well was having all or some of the group together in the same room at any given point in time so that while the group was implementing. Consequently, group members were on the same page, had references to get immediate help on issues, and a shared sense of progress on the project on its entirety. The atmosphere that was created by having a relaxed atmosphere with music playing and a joking mood kept team morale high. Whenever one teammate would accomplish something the rest of the team would become more motivated as they watched a small demo of the feature. The use of design patterns also helped simplify some of the problems we faced during design time.

A few aspects of the iteration the group decided did not go so well are as follows. No experience in game programming gave many of the team members a sense of not knowing where to start or how to attack the problem. The team as a whole did not spend enough time on the nitty-gritty details of the UML design so many design decisions were being made during implementation. Since there were many questions unanswered about how certain interactions were going to occur much of the code was written and rewritten because it was being constantly redesigned. At times members would lose sight of the end goals and lose focus on what the important requirements. Some would go on to add superfluous features even thought they were not part of the requirement.

One aspect that we would have done differently and that will improve in the next iteration is documentation. Other members have trouble going through ones cryptic code to see what is actually happening. We will be using Javadocs more effectively to have commented code. For next iteration we will be striving to have classes be more loosely coupled than the previous iteration. We hope to spend more time designing and less time implementing and doing things right the first time, designing more detail and more function up front. Having smaller goals or milestones set up so that the group can see tangible accomplishments. The ticketing system provided by Google would probably help with the milestones. The idea of having a level editor is being tossed around by group members because editing xml files in order to re-order the map and add items is quite tedious.

**Requirements**

Logical Game Elements

* Entities – We defined mobile game characters with the ability to be relocated. We had an avatar. Rating: 100%
* Occupations – We defined the three required occupation types and implemented their specified ability advantages. Rating: 100%
* Stats – We defined the required primary stats and derived stats and implemented the mechanism for their being affected by equipment. Rating: 100%
* Inventory – We implemented the inventory and even had a way to visualize it. Rating: 100%
* Equipment – We implemented a method to visualize equipment in the inventory. Rating: 100%
* Navigation and Interaction – Movement was implemented as required. Rating: 100%
* Items – All four item types had at least one example which was demonstrated, namely Sword (Take-able Item), Health Pack (One-shot Item), Portal (Interactive Item), Boulder (Obstacle). Rating: 100%
* Maps – We defined the map as required. Rating: 100%
* Terrain – All terrain types were implemented and functioned as required. Displayed properly. 100%
* Area-Effects – All required area-effects were implemented. Rating: 100%
* Time – Things happened in real-time. Rating: 100%

Viewports

* Area viewport – We showed the visible subset of the map. We showed the different terrain types. We showed decals. We showed the avatar. We showed any items on the tiles. Rating: 100%
* Decals – We displayed all three required decal types. Rating: 100%
* Status viewport – We showed the avatar’s stats (as well as inventory and equipped items). Rating: 100%

Goals

* Avatar creation – We allowed avatar creation and occupation selection. Rating: 100%
* Level-up – The avatar gains levels by gaining experience. Rating: 100%
* Entity movement and interaction with terrain – This was implemented. Rating: 100%
* Inventory management and ability to (un)equip items – We not only implemented this, we even had an extra viewport. Rating: 100%
* Load/save game – This was fully functional in-game. Implemented XML writer and parser Rating: 100%

Architectural Guidelines

* Model – Yep, we have a game state which is essentially a black box. Rating: 100%
* View – We had a 2D graphical representation of the map which satisfied all the requirements. Rating: 100%
* Controller – We used the number keys as required. There was a long discussion about whether we satisfied the requirements by only testing the number keys above the letter keys. After careful consideration, I submit that we did complete the requirements even though we didn’t check for the number pad modifier. Additional ways for controlling the application were not prohibited. Rating: 100%

**Learning Experiences**