Cleansing the Undead

Final Project Document

Jéssica Pauli de Castro Bonson (B00617515)
Pedro Elnízio Távora Pinho

CSCI4168: Game Design & Development - Devin Horsman
December 12, 2012

Dalhousie University

Index:

1. Introduction:

Cleansing the Undead is a 2D platform game with RPG elements, where a crusader is responsible to purge the undeads that are attacking an area and kill the necromancer who caused it. It was developed using Unity 3D, Photoshop and several game components from other games, specially from the game Ragnarok Online.

2. Game:

In Cleansing the Undead the player is Irene Rurcie (Fig. 1), a crusader who works for a religious organization that has the goal to clean the world from the dark forces. In its story, recently more places than usual have started to ask for help against undead monsters, and a more deeply investigation noticed that most of those places had been a war field. The organization is worried that someone is trying to ressurrect old armies around the world and is sending its soldiers to eliminate the undead creatures in the attacked places.

Figure 1. Irene Rurcie (main character)

Irene was sent to a forest festering with by undeads and her final goal is to kill the necromancer that is summoning them. There are three types of enemies, following the basic RPG classes: a melee, a ranged and a mage (Fig. 2). Each of them has it own, rule-based, AI: the melee patrols an area and chases the crusader if she goes near it, attacking her when possible (not in cooldown); the ranged patrols and area and shoot her when she is in its range; and the mage enemy has three different skills that it uses depending upon the situation, to damage her (thunderball) or to keep her away from it (hurricane and lightning). To all attacks and skills there are an specific cooldown which limitates how frequently the enemy can use it.

Figure 2. Melee Enemy, Ranged Enemy and Mage Enemy

The final boss has several skills: 1) Teleport: It's used to run away when the crusader tries to attack him, it's also used to position him to use some skills that require it; 2) Icicle: Ice stakes are randomly summoned in the ground, they last for some minutes and inflict damage over time the crusader touches them; 3) Curse: The crusader is affected by a curse which halves her speed and negates her hp and mana regenerations; 4) Blame: The necromancer teleports to the middle of the battlefield and invokes explosions around him, dealing a huge damage; 5) Summon Undead: Two undeads of one of the three types are summoned to help the necromancer; 6) Life Drain: When his HP is below 20%, the necromancer teleports near Irene and starts draining her life, this skills can be interrupted if he is hit by her. Figure 3 depicts the final boss.

Figure 3. Necromancer (final boss) standing and casting Curse.

3. Game Mechanics:

As the crusader the player can have basic horizontal and vertical movimentation using AWSD or the directional keys, they are also able to attack pressing E or J, and to defend pressing SPACE BAR or K. Defending cuts the damage from melee enemies by half, and nullifies the damage from ranged enemies.

The is also and level up and skills system. The player starts at level 1, the maximum level is 12. Each enemy killed grants some amount of experience, and after some experience she ups a level. The level up function is a altered version of the exponential Y = 155.4*1.287^X, it was altered so after killing all the the enemies the player would be at the maximum level.

At each level the player obtains one skill point, that can be used to activate or improve skills. Active skills can be used throught the numbers 1 to 4 in the keyboard, but there are also 2 passive skills. The first active skill is Heal, that heal the health of the crusader; the second one is Vengeance, and long-ranged wave which inflicts damage to the enemies is passes through; the third is Judgement, it's a high cost skill which deals a great damage in an explosions around the crusader; the last active skill is Intervention, it has the higher cost and after using it the crusader will be imunne to damage during some seconds. The passive skills increase the maximum HP and the attack damage of the crusader. It's relevant to note that the way we designed the skill Vengeance it ended up almost useless, since it needs a continuous terrain to be used, and after the Stage 1 we though that floating platforms would be more interesting and dynamic to the player. Figure 4 depicts the sprites of the active skills.

Figure 4. Active skills sprites (Heal, Vengeance, Judgement, Intervention)

The game has a basic HUD, which displays the HP, mana, level, skills and menu options, as shown in the Figure 5.

Figure 5. Game HUD

4. Game Design:

Most of the sprites used in the game are from the game Ragnarok Online, the other relevant references are in the References section at the end of this document. All sprites were adapted though Photoshop to suit better our game, like adding shadows, some corrections, improvements and creating the mirror image. And some of them for skills were created from scratch (ex.: Judgement, Intervention, Teleport). For most sprites we took between 1 to 3 hours to find a sprite that suits our game and to adapt it, it was way faster than creating all sprites from zero (those took 2 to 4 hours), but it also constrained us about what we could do in our game.

An example of constraints is about the boss sprites, initially we were planning him to walk around and cast some skills while animating his body, but his original animations for walking were quite bad, and even trying during some hours to create a new sprites for walking or casting skills, they didn't look nice, because of it the boss uses the same casting animation for all skills, and moves just using the skill Teleport.

5. Overview of the program code structure:

Because of the time constraint, we programmed the game in the way that we were used to with regular object-oriented programs: A class for each main concept, using inheritance to share common code. Also because of this constraint, we developed the game by improving a prototype.

Because of it our code ended up with very big classes for more complex concepts, like the player and the boss, each one with more than one thousand lines of code. It also was hard to reuse code, and so there are a considerable amount of copy-paste codes. The inheritance was used to develop the code for the three types of enemies, but it also ended up being not very reusable, since we couldn't use it to create the code for the boss.

If we would continue to develop the game Cleansing the Undead we would focus hardily in organizing the code in components with a smaller quantity of code, like creating a component "Take Damage from Crusader" that could be used to compose the three types of enemies and the boss.

The code also uses a lot of if-then-else to control the flow of the characters actions in relations to cooldowns, and because of it we have more than ten if-elses nested or in sequence to control the action flow of the player or the boss, which is quite hard to maintain. Ideally we should use coroutines to deal with this problem of synchronizing actions, animations and skills.

6. Time Management:
Before starting the game, we organized a simple schedule with what we should do in each sprint. It was followed successfully with just some minor changes, and it was as follows:

1 week: 1 stage, combat game mechanics without skills, dummy enemy
2 week: 1 stage, melee and ranged enemies
3 week: leveling up system, passive skills and active akills (1/2)
4 week: 2 stage, mage enemy and active skills (2/2)
5 week: 3 stage, boss
6 week: user interface
7 week: polish
8 week: polish

The changes were that the HUD was developed earlier in the sprints, and that basically everything about skills (crusader's, boss' and mage's skills) took a longer time than initially expected, because their sprites were hard to find or create, and they were also hard to implement, since each one had different special features, that most of time generated some bugs. Another thing that changed is that besides sprints 7 and 8 being dedicated to polishment, we also started polishing the game right after Sprint 2.
In our task sharing Jéssica was responsible for everything related to implementation and Pedro was responsible to most of the things related with game design (sprites, sound, music, level design). In the first two sprints Jéssica worked more, since the goal was to develop a simple working prototype, but after if we usually got and equal work load. It was a nice way to share the tasks, because we were able to focus and get better at different parts of the game, and because of it we improved and worked quite fast.
Our task management chart (Fig. 6, where blue is the expected time and orange is the actual time) linearly increased along the time in regards of the expected time, it happened because after the initial sprints Pedro were able to work more in tasks regarding polish, because Jéssica dropped a class around Sprint 4 and because we noticed that developing skills was time-consuming. The orange line, the actual worked time, reflects that during all the project we tried to find a balance between working too much and doing enough work, besides working a lot of extra time in some sprints, we didn't do it consecutively. For Sprint 8 we expected to just have to deal with some polishment and small bugs, it was accomplished and so we finalized the game calmly.

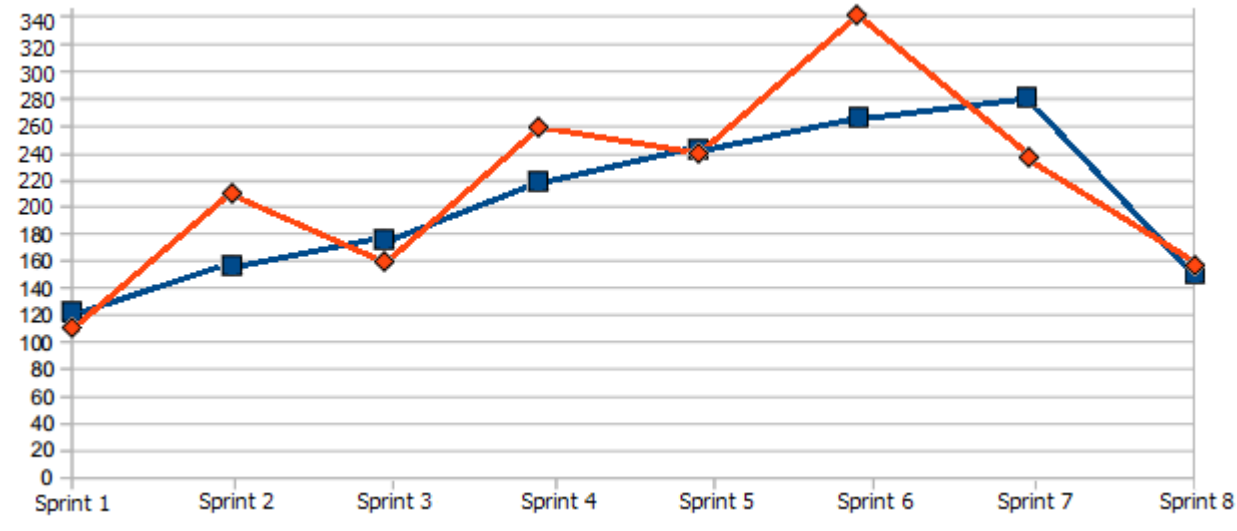

Figure 6. Time management chart

7. Lessons Learned:
About the code structure, as we said the section 5, we learned that to develop a game using Unity the code is much more reusable if it's strutured around components and composition, instead of complex classes and inheritance. And it's probably better to synchronize animations, actions, sounds and skills using coroutines instead of if-then-else's, since the last one produced a code hard to manage. We found easier to synchronize the game based upon real time passed than by frames, and also we developed the code for the boss based on real time instead of frames.
For a future game, we would try to avoid using Sprite Manager 2, since because of it we got stuck in details about atlases, and we got some bugs regarding it, some of them were solved just when we deleted everything and tried again to generate the atlases again.
Finding, editing and, in a few times, creating sprites it was already quite time consuming, around 1/3 of our time developing the game was spent in these activities, and much more time would be spent if we tried to create all sprites from scratch. Because of it, if we would develop another game with quite complex movements for the characters and with skills, having enough time for it, we would prefer to work with 3D animations, since we would work a lot to create the model, but after it the creation of animations, effects and movements would be faster.

8. Conclusion:
We learned a lot developing the game Cleansing the Undead and we are very satisfied with the result, since it's a game that ourselves and most of our friends found very fun to play. We found that the lessons we learned are going to be very useful to develop other games in the future.

References:
- Ragnarok Online (sprites, sounds, musics, start menu image and some skills)
- DeviantArt @admin2gd1 (background image)
- Pixelation @Gromit (textures for final arena)
- King of Fighters 97 & 2000 (boss skills "Blame" & "Icicle")
- Megaman X4 (mage skill "Thunderball")
- My Pet Protector 3 (skill icons)
- Mirrors Edge (victory music)
- Pokemon (sign at the end of levels)