UNIVERSITY OF MALTA, INSTITUTE OF DIGITAL GAMES

Prototyping for Game Designers

Game Report — RPG: Real Plain Game

Jean-Luc Portelli

February 2, 2014

1 INTRODUCTION

In this report, I will be speaking about the development and results of the experimental role playing game named RPG: Real Plain Game. I will be dealing with the less obvious aspects of its design, the challenges faced within its implementation, as well as the results shown as part of the experiment.

2 DESIGN

Real Plain Game aims to challenge, or at least bring to light, the main gameplay tropes which the vast majority of classic and modern role playing games' gameplay is strongly based upon; specifically, those of levelling up and grinding.

2.1 "LEVELLING UP" AND "GRINDING" AS MECHANICS IN RPGS

In most JRPGs, players are tasked with embarking on a long journey as a party of characters. As they play, they are frequently engaged in combat with monsters, trading damage back and forth based on the statistics of their party. These statistics affect the strength of the characters' attacks, as well as how much damage they take; in other words, these stats are what the vast majority of the game's progression is based upon.

One major aspect of this system is that, as you fight, you receive experience points that increase the statistics of your party—known as "levelling up"—which, in turn, improves their strength and makes it possible to dispatch more and stronger enemies. Hence, as long as you are able to defeat something, you will grow stronger, and eventually progress to a new area

that may be populated with stronger enemies — enemies that you may not have initially been able to beat.

One possible strategy that can be applied to these kinds of games is that of "grinding;" namely, repeatedly defeating weaker opponents to raise your stats in order to progress more easily. This is often time consuming, but it effectively trivializes the later challenges in the game: the side effect being that, given enough time and patience, a player may reach a level of strength that is far greater than what the game designers had intended.

In other words, *most of the challenges within RPGs can always be solved by means of grinding.* Many lament that it is an unsatisfying way to play the game; however, even if a player refrains from grinding deliberately, the vast majority of the game is dominated by this process of constantly levelling up.

There are some games that tackle this issue by means of either reducing the effectiveness of levelling from weak enemies (Final Fantasy Tactics), or completely removing the points earned in an area after the player reaches a certain level (Mario & Luigi games). Point scaling is a common trope in RPGs where designers want to discourage grinding[2].

Within one such game, called Bravely Default (the main inspiration for this experimental game), you take on the role of a typical band of allies involved in a typical world-saving plot. What makes this game particularly relevant here is that the player is allowed to, at any point and as many times as they wish, change both the difficulty as well as how often enemies are encountered. In addition, there is a feature that allows the player to pre-program commands for the characters to carry out at the start of every battle.

As a result, one may adjust these settings in order to grind without even having to actively play the game, effectively excising the process of levelling up from the experience.

Even more interestingly, within the same game, it is also entirely possible to play the game without even having a single random encounter[1]. This is the essence of the concept we are interested in.

2.2 Real Plain Game's Experiment

RPG takes this concept even further, to an extreme; where the whole process of levelling up is removed and placed entirely in the player's control. Grinding is not so much unnecessary as much as it is impossible to do.

The idea behind this change is to consider two primary aspects that reflect on the genre as a whole:

1. What is left of the game once the *process* of levelling up is removed?

2. What gameplay and design considerations emerge as a result of this change?

Achieving this would involve letting the player level up their party of characters on command, literally choosing the level they wish to have. The level still determines the party's strength and ability; however, in-game progression is measured only through solving challenges in the form of important fights and possibly even puzzles.

The repetitive process of levelling in role playing games comprises a large chunk of game time in most RPGs. Expectably, the removal of such a prevalent portion of the game may

result in a more abridged experience. However, whether or not players get the *intended* experience will still rely on their choice to be at the appropriate level or not.

3 TECHNICAL

When developing the major portions of *RPG*, there were various technical considerations to be taken for its implementation. With regards to Game Maker, the tools it provides for game design bring many technical hurdles.

As part of the experiment in this game's implementation, I also decided that the game should be designed without the use of GML in any way. This is mainly just to see if it is possible, but also what, if any, added complications would surface in its implementation as a result.

As far as the game's features are concerned, a number of decisions were taken right at the start of the design process. This was in order to narrow down what the game, as a functional RPG, should consist of aside from the main mechanic — or lack thereof.

1. The player controls a party of four characters with specific, differing roles
2. The game allows for exploration around a worldmap
3. Combat would always be voluntary; there are no random encounters
4. The party's level is shared; no characters' level is higher than any other
5. Combat will only ever occur with one opponent at a time
6. Each character will take one action out of two choices before the enemy attacks

These considerations covered the bulk of the game's design. Some of them may have been thought up along the course of the first stages of development; however they were set in stone fairly early on, for purposes of simplicity or to keep the implementation fairly tractible.

The first part of the implementation was the world map. This was done using a tile based background, but in order to restrict movement, abstract colliders were placed where the player was not meant to be able to move. Movement was also grid-based.

The first challenge faced, compounded by the unavailability of GML, was for allowing interaction with objects that the player character was facing in the world map. This was the first problem of many that would be dealt with using a common solution within *RPG*.

3.1 INTERACTIONS BETWEEN GAME OBJECTS

One of the biggest restrictions within Game Maker is that objects within the play area have no way of communicating events with one another. While it is possible to use a central game controller, it is difficult to know exactly which object should be receiving the data stored within the controller. Even at that point, it is even more complex to allow for two-way communication between the concerned objects.

Encountering this problem within the context of interaction on the world map resulted in a simple solution that would be used across the entire game. One of the simplest ways for objects to communicate with one another is through collisions; once two objects are involved in a collision, they afford full, unambiguous visibility to one another. Even within the restricted domain of Game Maker's event driven programming system, it is provided through the use of the "other" property within the collision event.

When put into practice; in the case of interacting with an object on the world map, this is achieved by first placing an invisible object on the world map's grid. This object is a child object to the obstacle mentioned earlier, but it also possesses an action it must carry out. In the general case, these objects would display a message on screen to the player, imparting information or otherwise. If they interact with an enemy that is lurking on the map, it begins a fight; but that will be discussed later.

Whenever they wish to interact with objects on the map, the player presses Space. This creates a pixel-sized, invisible object in the middle of the grid cell in front of the character, which only lives for 2 time units in-game before deleting itself. This object is what I refer to as a collider.

Once the collision is registered by the receiving object, if any are present in that space, the relevant action is triggered and the collider deleted. You could compare the behaviour as a method call, where the *type* of collider used calls a specific method of the object being collided with.

For the sake of brevity, I will discuss the most notable challenges that colliders helped solve, and how. It should be mentioned at this point that these *could* be solved through other means, however, as they say: *when all you have is a hammer, everything looks like a nail.*

3.2 Exchanging Damage Between Fighters

For this problem, collisions make the task of exchanging damage (in the form of numerical values) fairly trivial.

As it is implemented, the object dealing the damage sets a global variable in the battle controller. It then creates a damage-type collider on the target, which then receives the value stored in the battle controller.

Following that, the resultant damage is calculated based on the defense attribute of the target, adjusted accordingly, and deducted from the hit points. For the sake of feedback, the amount of damage dealt is shown as an object which draws the value for a brief period, with an upward motion and gravity set to give a nice "jumping" effect.

3.3 Picking a Valid Target for the Enemy

This was a far more complex issue, caused by the fact that the battle controller was not in charge of maintaining each object's hit points. While randomization is key in picking a target, of course, an enemy should not attack a character that is already at zero HP.

In order to check which targets are valid, I set up an iterative loop through the use of alarms, where the following loop takes place:

1. Enemy creates "status" collider on next character

2. When character meets "status" collider, it uploads its current HP to the battle controller

3. At start of next tick, Enemy checks HP value in battle controller and if HP > 0, it increments number of valid targets and adds its position to an array

Once the loop checks all four characters, the enemy randomizes based on the values in the array. That way, only the positions of those characters can be picked as possible targets.

3.4 Turn Based Actions

This problem's solution ended up being fairly elegant through the use of colliders. Essentially, the turn order of combat can be seen as a state machine, and colliders are what trigger the transition from one state to the next.

Every object in the battle (the characters and the enemy) is either active, or it is not. Through the use of the "turn" collider, the objects effectively play "pass the parcel" where, whenever an object meets the collider, one of two things happen:

1. If inactive, become active and resolve an action
    - When action is resolved, create a "turn" collider on self

2. If active, become inactive and make a "turn" collider on the next character in order

In the case of characters, action resolution takes place as soon as the player chooses an action. On the other hand, the enemy being fought automatically carries out its action and passes the turn to the first character in order.

4 Evaluation

Two playtest sessions were conducted for *RPG*, following its completion. In the two cases, the resultant gameplay styles were diametrically opposed to each other. In one, the player was extremely conservative with levelling up their party, while the other player brought it up to the maximum level as soon as it was possible.

In both cases, we see two very different outcomes in the priorities of play, however they still both had a considerable amount of overlap.

4.1 Player I: The Combat Specialist

The first playtest was characterized by calculated levelling of the party, and planned combat.

During gameplay, a lot of attention was given by the player to the combat encounters and level design. They chose to avoid combat wherever possible, only levelling up as necessary. Due to the way certain enemies were positioned, this led to an unintentionally steep difficulty curve that stalled the experience at the mid-way point. The player ended up a tad frustrated at having to use trial and error quite extensively, to try and guess the right level needed for the challenge in fights that they sought after.

In spite of this, they persevered, and eventually completed the game at a far lower level than was initially anticipated as a minimum requirement. In the final battle, which had a small puzzle incorporated into it, the player did notice that a pattern was present. However, mainly due to the lack of conveyance in the game (not enough visual or audio feedback), they succeeded without recognizing the exact pattern.

Following the playtest, the player suggested a lot of design considerations related to these two main experiences. Firstly, they observed that such a game would greatly benefit from content surrounding the mechanic of combat; namely, item rewards for progression or altering the aesthetic of the game, an improved storyline, as well as allowing flexibility in how levelling should be handled. For example, the game could automatically bring the player's party up to a certain level at specific points in the game to match the difficulty chosen. Additionally, they noted that better designed maps would be necessary to ensure the players do not have a jarring shift in difficulty without their knowledge.

4.2 Player II: The Completionist

As mentioned, the second player went down a completely different route, and maxed out the level of their party as soon as they could. This inevitably led to the encounters being trivialized outright, however, the experience shifted to a different priority as soon as combat was no longer an issue.

Within the small map, several hidden points of interest could be found by the player, leading to them exploring every nook and cranny that they could. In this way, the gap left behind by the combat encounters were summarily filled by flavour text littered around the world; either in landmarks on the map, or whenever the monsters are fought and dispatched.

This polar shift in priorities highlights the necessity to have other activities to back up the game. People who would choose to blaze through a game like this would be interested in the progression of the story and to see the world that the game designers created. In short, as the player expressed unequivocally, they would prioritize exploration above all else. Even after claiming that they spend considerable amounts of time grinding in order to achieve greater levels of strength in the RPGs that they play, if given the option, they would prefer to play an RPG that prioritizes exploration over grinding, even if it is otherwise identical.

5 Design Observations and Conclusions

When speaking about the game's experiment's results, it is important to note that some observations were made even prior to the game's completion, in the process of its design.

The most important observation was the way the game's *content* would have to be considered. Since the process of levelling is absent, it would make for a much shorter game than if it was otherwise included. Had it been developed as a fully-fledged game, other features would need to be incorporated to compensate somehow, ideally to increase its engagement more so than the amount of time it consumes.

Instead of focusing on designing countless smaller monsters to fill in the time for grinding and levelling, the best alternative would be to have fewer, but far better designed combat

encounters. These encounters could easily incorporate some other kinds of mechanics, in which combat would need a form of problem solving, over and above (or even instead of) the usual rigamarole of dealing damage while keeping your party healthy.

While *RPG* goes about the removal of levelling in a fairly clunky way, literally adding a level with each button press, this mechanic can be designed better around the game's intended progression. For instance; each battle, although voluntary on the part of the player, could grant the player specific rewards and a certain number of levels upon completion.

This would allow, based on the intended difficulty of each battle, the designers to better control the progression of the game without necessarily letting the players break it entirely. Creating a limited resource out of combat encounters, and what rewards they grant the player, would better control the flow of gameplay within a more manageable environment.

Another avenue that can be explored is tying in combat rewards with other tasks within the game. Perhaps combat would be a form of resource collection for some other, entirely different part of the game. It was also discussed that optional encounters would yield rewards in the form of cosmetic items, since items as strengthening equipment are rendered redundant.

During discussions, both playtesters expressed an interest in exploring the world, or otherwise being involved in more than just combat. With far less emphasis on it, or at least, making it less time consuming, the game can open up to new avenues.

One such avenue is the *actual role playing* part of the genre that is, quite ironically, sorely absent in so many of them.

REFERENCES

[1] You can complete bravely default without random encounters. URL http://tinyurl.com/nf4og2w.

[2] Main/Anti-Grinding - television tropes & idioms. URL http://tinyurl.com/ot62ors.

RESOURCES USED FOR DEVELOPMENT

[1] Pixelized 'The NorthEast View of Edinburgh Castle.' Used for intro.
URL http://digital.nls.uk/slezer/engraving.cfm?sl=59

[2] 'Pixel art sunset.' Edited and used for epilogue.
URL http://joazzz.deviantart.com/art/Pixel-art-sunset-134221602

[3] Free flash pixel fonts. Used for in-game text.
URL http://www.thepixelplant.net/npp/free.asp

[4] as3sfxr — 8-bit sound effects generator. Used for sound effects.
URL http://www.superflashbros.net/as3sfxr/

[5] NES Dragon Quest OST. Used for game music.
URL http://www.woodus.com/den/music/mp3.php?desc=dq1nes

[6] 1-Bit tileset. Used for world map tiles.
URL http://opengameart.org/content/tileset-1bit-color

[7] Final Fantasy Adventure spritesheet. Used for many in-game sprites.
URL http://www.fantasyanime.com/mana/ffadventsprites.htm