Implementing new mechanics on a motion game to achieve good player experience

Autor: Mario Pizcueta Martínez
Supervisora: Aseel Berglund

Linköping, JUNIO 2016
Index

Abstract................................................................................................................................. 3
Resumen ................................................................................................................................. 3
1. Introduction ......................................................................................................................... 4
  1.1 Description and goals of the project.......................................................................... 4
  1.2 Chosen mechanics to implement .......................................................................... 5
2. Development ......................................................................................................................... 5
  2.1 Description of original project status ................................................................. 5
  2.2 Original design of game mechanics ................................................................. 6
  2.3 First game theme: Night ...................................................................................... 6
    2.3.1 Description ...................................................................................................... 6
    2.3.2 Development .................................................................................................. 7
    2.3.3 Encountered problems ................................................................................. 8
  2.4 Second game theme: Winter .................................................................................. 8
    2.4.1 Description ...................................................................................................... 8
    2.4.2 Development .................................................................................................. 9
    2.4.3 Encountered problems ................................................................................. 10
  2.3 Third game theme: Spring .................................................................................. 11
    2.3.1 Description ...................................................................................................... 11
    2.3.2 Development .................................................................................................. 12
    2.3.3 Encountered problems ................................................................................. 13
  2.4 Other miscellaneous work ................................................................................... 13
3. Game experience test ........................................................................................................ 13
  3.1 Description of the process .................................................................................... 13
  3.2 Obtained data ......................................................................................................... 14
4. Final conclusions ............................................................................................................... 15
Abstract

The game industry is growing fast and the game mechanics available now won’t be enough to sustain this growth. Because of this, the need of new mechanics is essential to the game industry. The goal of this project is to design, develop and implement new kinds of game mechanics for a motion game under development on the Phaser game engine, under the requirements of the game designers and working together with a team of developers working on the same project with different goals. This game mechanics are divided into three different groups or themes inside the game, and on each theme there are several different mechanics implemented, and all of them are the result of hard development with many difficulties to overcome. Once all the mechanics are implemented, the new mechanics were tested for user experience following common testing methods.

Resumen

La industria de los videojuegos está creciendo rápidamente y las mecánicas de juego disponibles hasta el momento no serán suficiente para soportar este crecimiento. Debido a esto, la necesidad de nuevos tipos de mecánicas de juego es esencial para la industria del videojuego. El objetivo de este proyecto es diseñar, desarrollar e implementar nuevas mecánicas en un juego de control por movimiento en desarrollo, respetando los requisitos de los diseñadores de juego y junto a un equipo de desarrolladores trabajando en el mismo proyecto pero con diferentes objetivos. Estas mecánicas han sido divididas en tres grupos diferentes o temas dentro del juego y en cada uno de estos temas hay diversas mecánicas de juego implementadas, todas ellas fruto de un duro desarrollo. Una vez que todas las mecánicas han sido implementadas, estas han sido sometidas a test siguiendo métodos de testeo comunes.
1. Introduction

1.1 Description and Goals of the project

The video game industry has been experimenting a great growth during the last few years and it is expected to keep its growth speed for many years to come. However, the overuse of already existent game mechanics is creating a problem when developing new and fresh content on forthcoming videogames, especially when it comes to the new technologies that have been emerging lately into this industry such as VR or AR. That’s why this project goal is to design and implement new game mechanics into an existent motion game, providing a good game experience to the player.

To achieve this, I will be working with the videogame development framework Phaser, which provides a complete toolset for continuing developing the motion game. I will also be working in a team of three workmates, each of us with different task but with the same goal of improving the motion game.

The videogame in which I will work is a motion strategy game where your main task is to protect trees against walking bugs that are trying to eat it, while achieving the maximum punctuation possible. The game is until now divided into eight chapters and each chapter is, at the same time, divided into ten different levels. On each of this levels there is a tree to be defended and the player must achieve the maximum punctuation, represented by a maximum of three glowing trees, by squishing the coming bugs and avoiding the water drops that are beneficial for the tree. While the player advances in the game, there will be appearing different kinds of bugs and other elements to alter the game and provide variation and progression.

The game is a motion-controlled game, it means the player interacts through body movements and the input is mostly provided by gesture recognition. The wide developments of this kind of game is relatively new, mostly pushed by the launch of Kinect on November 2010, it can be consider by some as one of the new technologies of the industry as it is now being integrated in many aspects of the augmented reality.
1.2 Chosen mechanics to implement

For this project I will select a wide range of game mechanics to implement. This game mechanics must fit the already given basic gameplay and must be designed according to the porpoises and prerequisites given for the game by the original designers. Once implemented this new mechanics, they will be subjected to a general and basic user experience test to study if this mechanics have fulfilled their goal of providing a good experience to the player.

I tried to keep the set of mechanics to implement as wide, diverse and scalable as possible so the player can really feel the variation and progression while the game is evolving. The selected mechanics are divided into three different game areas or “themes” according to the game structure. This themes will be named “Night Theme”, “Winter Theme” and “Spring Theme”, and each of them will provide something new to the normal game or “Standard Theme”.

Every theme to be implemented includes several new mechanics, some of them affect the general gameplay and some are merely noticed. This mechanics will be detailed on forthcoming sections.

2. Development

2.1 Description of original project status

The project was originally developed on the Phaser engine as part of another student’s final project. The game already had a general file structure, provisional sprite models and the basic core mechanic implemented. It also had a procedural generator of level paths used for creating 100 levels inside the game divided in 10 different chapters. This
levels wasn’t created dynamically inside the game, but they were previously created under supervision and stored inside a file for using them on the game.

On this original game, the core mechanic was based on a displayer on the top of the screen showing which kind of bug the player have to squish next, and until exactly that bug was squished, the player were not able to squish any other bug in the game. This mechanic was later proved to be hard to understand and complicated to play.

The game had a rotation system among 4 different seasons (spring, summer, autumn and winter) that provided visual changes on the sprites during the game. Besides this visual change variation, the game had very few implemented variation mechanics such us flying bug that jumps after they are squished once and ice bugs that slows the rest of bugs and water drops after being squished. This game variations were not enough for the amount of levels available, and after playing the game for a while it became monotonous.

2.2 Original design of game mechanics

After playing the game several times and getting used to the code, it was clear that the best approach for adding new content to the game that provide more variation was creating different game modes or “themes”, and inside each of this themes adding and changing all the mechanics. Every theme was supposed to have a core game mechanic characteristic of the theme, at least one special bug with special abilities related to the theme, and different small mechanic variations for improving the experience. All of this combined with general visual changes on the sprites.

The idea of the themes was provided but an already implemented small variation on the game called “the dessert theme”. Although this theme was still vague and didn’t had any file structure supporting it, it was a great starting point.

2.3 First game theme: Night

2.3.1 Description

The first game theme to be implemented is going to be light based. It means that the core mechanic of this themes will be light changes that prevent the player from seeing most of the map, and every bug would be provided with one personal light that illuminates its position and direction. Also the water drops are also provided with light to avoid squashing them unintentionally.

The idea behind this mechanic is to add some fuzziness on the map so the player is not able to see or memorize all the routes of the bugs. This means that the player don’t know where the bug is going to appear, and where is it going to go, differing from the standard game where spawning points and the routes are plain clear.
2.3.2 Development

At the beginning of the implementation the idea of how to achieve this mechanic wasn’t clear. The first prototype function was in the Game.js file, as all the other special mechanics were by that time implemented in this class, and it was called from the update function provided by the Phaser engine. In this first function I started using the alpha of the sprites to create the light play. Instead of hiding and showing the map depending on the light, the bugs were the ones who were hidden and shown.

This first implementation was discarded by the game designers mainly because it was hard to understand why the bugs where hiding and showing and could generate frustration on the player.

For the second implementation of the mechanic I started using the bitmapData object over a sprite of the size of the screen and using Phaser.blendModes.MULTIPLY to multiply all the sprites of the game under this sprite and this way achieving the darkness effect. To set the blend mode as multiply was necessary so later different forms could be drawn over this sprite showing the real game beneath. The setup of this sprite was originally on a function inside the Game.js file and it was called at the beginning of the game level. As the bugs appears, another method was called from the update function. This function was in charge of drawing the different light forms over the bugs and the water drops. It mainly consisted on a loop iterating over all the entities of the game and depending drawing the light setting the alpha to 0 and a gradient so it has the desired light effect.
This second implementation was quite satisfying. Later on there were some modification rewarding on the shape of the lights to get a nice looking view, but the core mechanic was already implemented.

After the approval of the game designers I moved on implementing the special bug for the theme. This special bug consisted on a big light source big enough so the player found it convenient to keep in the game as long as possible. This way it will be a bug the player actually won’t want to squish, still being a bug.

For the implementation of this bug I used the Bug class, extending from the PathFollower class, where all kinds of bugs are defined, to define a new kind of special bug. Later on, from the function previously implemented controlling the light sources, I only had to extend the light source of the bug in case is one of this special bugs. After finished implementing this special bug for the Night theme, this theme was almost ready to be tested and polished, and its code remained immutable until the file structure change explained later on.

2.3.3 Encountered problems

After the implementation, the main problems rewarding to this new theme where related to the performance. As the sprite was present during the hole theme and the update function was continuously redrawing this sprite in different shapes, the FPSs of the game where slightly affected. With the help of one game designer we get to the conclusion that using shaders instead of a bitmapData sprite will remove any possible performance problem, and this could be the best approach for future works on this theme.

2.4 Second game theme: Winter

2.4.1 Description

The second theme to be implemented was the winter theme. This theme was meant to use the same graphics as the winter season on the Standard theme, adding some changes on the mechanics. The main different is the falling snow. During all the levels there will be falling snow that covers piles up in the bottom of the screen and in different parts of the roads. This mechanics is designed to hide the paths and elements of the game and also to create distraction to the player by adding tones of movement through all the screen.

This basic theme mechanic will be supported by the special bug, a frozen bug you must squish twice in order to kill it, and the slippery roads, frozen sections of the paths where the bugs highly multiply their speed. Later on some more mechanics were also added to this theme by one of my work mates such as the chili collectable that also increases the bug speed. (Collectables are elements that appear outside the paths and the player can collect).
2.4.2 Development

At the beginning of the development of this theme, one of my work mates implemented a new file system for the whole project. This new file system mainly added more cleanliness and a lot of scalability, as the original game code wasn’t prepared to be scaled that much. Also this new structure improved a lot the performance of the overall game. One of the main things of this new structure was that all the theme related mechanics and effects, instead of being on the Game.js class now they have their own class, one different class for each theme to implement that are directly instanced and called from the main game. For example this theme had its own file named WinterTheme.js. This were great and after the file structure was implemented, the code of the new features were much easy to implement.

Rewarding to the theme, the first element to implement was the falling snow. At the beginning I tried an implementation using again de bitmapData and dynamically draw the snowflakes over the screen, calling a method on the update function to rearrange the snowflakes on every iteration. It worked fine but the performance was catastrophic and the game was unplayable, so instead of using the bitmapData I changed the direction and I tried using a ParticleSystem for the effect. It worked fine and the performance didn’t seem to be highly affected, but the snow waves were quite unreal and didn’t look good enough. Finally instead of using only one particle system, I decided to use three different particle system throwing particles at three different speeds. This worked very well and didn’t affect performance that much.

After having the snow set up I started working on the staking snow. For this matter I decided to use the physic system of Phaser. The idea was to add physic collision to all
the particles thrown by the particle system, so they would stack above the different colliders set up on the bottom of the screen and randomly distributed in among the paths. This worked really well, and although the stacking wasn’t really high at the bottom. Finally I had to add two more long colliders on the side of the screen to avoid the snow falling from the sides. The physics made it look very real.

For adding variation among the different levels I set up the amount of snow and its staking speed depending on the level its being played. This way the first levels have very little snow and it is almost no disturbing, and the last levels have tones of snow making them more complicated.

After having the snow set up I started creating the slippery road. For this mechanic I had to change the map creation system MapGenerator.js to add a new kind of tile that was frozen. Also I changed the path keys already generated and I codified some of the tiles as frozen tiles. Then, once the paths are set up, all the Bug objects will have a list of all the frozen tiles inside the map, and every time the update function is called, this one will check if the bug is inside a frozen tile. If that’s the case, the normal speed of the bug will be multiplied by 4, and the rotation will be continuously changed trough tweens, small animation system of Phaser, so it gives visual feedback to the player that the bug is faster and sliding, all of this placed inside the function isOnIce() of the Bug object.

Finally, the last thing to add on this theme was a special bug, so I started implementing the frozen bug. This bug was supposed to be inside a frozen cubit, so the first time it’s squished, the cubit will disappear leaving the bug immune for some seconds and will expose the normal bug. Now the player will have to squish it again in order to kill it. For the implementation of this mechanic first I tried using the remains of an old mechanic called shield. It was supposed to work similar to the mechanic I wanted to have, but I didn’t manage to have this function running so I decided to implement my own mechanic from scratch.

The implementation of this mechanic was made on the Bug.js class, where a new type of bug was added and a Boolean indicator setting if the bug is frozen. Once the bug is squished, calling the squishBug() function on the Game.js class, this function will check if the bug is frozen or not. If is not frozen it will act like a normal bug, but if it’s still frozen, a function on the Bug.js will be called removing the frozen cube and making the bug invulnerable for 2.5 seconds. This time of invulnerability was necessary to prevent the player directly squishing the bug just by passing the hand once over it, as the motion detection will call the squish function multiple times until the bug is declared dead. For visual feedback on the frozen bug I added a provisional sprite of an ice cube, and for the invulnerability feedback I used the alpha on the sprite.

2.4.3 Encountered problems

The main problem encountered in these theme was again the performance rewarding to the snow particle system. As every particle of the system had its own physic collider, the number of calculations performed by the physics system of Phaser was too high,
especially on the latest levels, as the snow piled up faster. Also, as the particles piled up on the screen and never disappeared, the number of particles is always increasing as long as the player stays in the game. This performance problem was a little bit more noticeable that the one in the Night theme, as here at the end of the last levels the player can feel the FPS and the general game slowing down. One of the solutions to this problem could be using Shaders again, instead of normal sprites on the particle system, although I don’t know how much this solution could improve the performance.

Another problem is that the slippery road tile positions are coded inside the map keys and are recognized and display by the MapGenerator.js but cannot be procedurally generated in case we want to add more levels. The solution would be changing the procedural generation system of the game to add this kind of tiles in the winter theme.

2.5 Third game theme: Spring

2.5.1 Description

The third theme to be implemented was the Spring theme. For this theme we wanted a core mechanic that was completely different from the others, and after some thinking and brainstorming with the supervisor and workmates we came with the idea of the path builders. The mechanic consists of not having any path at all at the beginning of the level, and as the bugs advance, the build their own path. On the original idea, if the player plays the level well enough, the paths were never completely build until the tree.

For the special bug in this level we came up with two designs. One bug, more related to the core mechanic, will destroy the path created by the other bugs, making the other bugs to build it again. This would keep the construction and destruction of paths active, and the map very dynamic. The second bug would be the “Fire bug”, a bug that, once squished, explodes squishing and destroying all bugs and water drops around it. The idea is to make the player conscious and make him choose the best time for squishing the bug without squishing any water drop.

Finally, for this theme we thought on creating a collectible “the bomb” that follows the same mechanic as the fire bug but in a static way.
2.5.2 Development

At the start of the development of this new theme there were new structural modifications on the project, especially on the graphical part. Now the graphics, instead of being handled by their own objects as before, had its own class associated to the object. The change was useful but it took a little to get used to it.

The main challenge for this theme was the path creation and destruction. At the beginning I started with an implementation where the paths were dynamically drawn by the bugs using the `bitmapData` of a sprite. This implementation was simple, and components were very similar to the night theme, but the problem was that the paths were drawn very simple as it was a simple line over a plane background.

For the second implementation I decided to take the long way. The idea was to store all the graphic part of the paths into big arrays referring to each of the paths on the `mapGenerator`, and when a new bug was spawned, this bug will get those list depending on the path he was created on. As the bug moves forward, it starts changing the visibility of the tiles he moves over. On the start of this implementation I added an alpha twin animation to the appearing and disappearing tiles so it look more smooth, but along with the game designers we decided not to include the animation as it seems better without it. Once the creation of the paths was already implemented, implementing the special bug that destroys the path was really simple. I only had to slightly modify the bug class to declare the new special bug, and in case is this bug instead of a normal bug, the visibility effect of the path is reversed.

Now it was time to create the firebug. Once again for this bug, as it was a special bug, I had to declare it on the bug class. I also created the function called when the bug is squished. The idea of this function is, when the bug is squished, this will receive from the game class a list of all the entities available inside the game (all the bugs and the water drops). Once the bug receives this lists, iterates trough it checking de
between its position and the rest of the entities. In case this distance is smaller than the explosion range, the bug or water drop is squished. For the animation effect I used the `bitmapData` to draw the explosion range as a red circle, and through tween animations I made it scale and disappear as an explosion. This could be used as provisional range indicator until a proper sprite is added.

Finally, for the bomb collectable I used the same structure of all collectables implemented by one of my work mates and, as it follows the same mechanic as the fire bug but in a static way, I used exactly the same function into the collectable class. Finally I added a provisional bomb sprite.

### 2.5.3 Encountered problems

The main problem encountered on the implementation of these theme was to find a way of making it changing the least amount of classes available at the moment. I had a lot of ideas on how to implement the path creation and destruction mechanic but most of them involved a great amount of class, and for future development it would have been a nightmare to track all the implementation. The final solution is the one that involved the minimum number of classes possible and still created the desired effect. Besides that, the rest of the implementation went fairly smooth.

### 2.6 Other miscellaneous work

Besides working on the main topic of my project, I was also involved in the full development of the game working on different aspects not so close to my goals. This involves the improvement of the menus adding different animations where bug walk around the menus, the improvement of the water drops and modifying their sprite for adding animation a changing the into water buckets which we think provided more moral responsibility to the player to “save” the buckets as they are humanized, and other small works related to the graphical part, provisional sprites and creation of sprite atlas.

### 3. Game experience test

#### 3.1 Description of the process

Once all the new mechanics were properly implemented, it was time to make some testing for having some kind of feedback on this new parts of the game. As the tests was going to be made by myself alone as tester, the process couldn’t be very exhaustive and complicated. For this test I choose 5 volunteers with similar background. Each of this volunteers followed the same testing process. At the beginning, I explained the situation and I gave them some idea of what the game consist on so they know what to expect. After that they played around 5 to 7 levels including the tutorial levels on the standard theme, so they get to know the game without experiencing yet the new added mechanics.
After they feel comfortable with the game and its basic mechanic, they started playing the night theme. They played from 6 to 10 levels of every theme distributed among all the chapters so they can also feel the variation and progression on accelerated way. While they were playing, I used the think-aloud method to gather as much information as possible about the game and their feelings when playing it. Once experienced the new mechanic enough to be evaluated, I asked some open questions as to get them explain if they liked or not this mechanic and the way of playing it.

This process was repeated for each of the 3 new implemented mechanics and for all the 5 test volunteers.

3.2 Obtained data

All of the data obtained in this test is in the form of comments and impressions from the testers and its enough to have a rough evaluation on the implemented mechanics. Some of the comments on the first level of the tests of the night mechanic are about confusion and not understanding completely what was happening. This fuss last only for the first level they play, and once they get used to the mechanic it doesn’t seem much difficult than the standard theme.

The experience on the winter theme was slightly different. As these theme has more progression as the game evolves, on the first levels they almost noticed no difference. There is only some little falling snow and the special bugs doesn’t appear as well as the slippery road. But as the game goes further, they seem to get more stressed about the different components that appear. First the ice bug created some frustration as they don’t understand well that they had to squish it twice, and on the first levels where this bug appears some testers fail the level and have to retry. Once they realize about the mechanic of the frozen bug and they manage to get through this levels, then the slippery road starts appearing creating even more stress and confusion. Obviously these impression would be probably softer if they played more levels and had more time to adapt to every mechanic as it would be on the normal game mode, but according to the testers this theme was the harder to play although none of them refer as the theme as annoying or frustrating or mention having a bad experience.

Finally, the beginning of the experience on the spring theme was similar to the night theme. When they see the map without any paths takes a little time until they realize what is happening and what the mechanic is about, but once they understand it is not hard to play it. Also, when the fire bugs and the bomb collectable appear, they don’t pay much attention to them and even one of the testers don’t realize this mechanic until the last level he plays. This is probably because this mechanic is understandable only if there are some bugs or water drops near the firebug or bomb, otherwise is hard to understand.
4. Final conclusions

To conclude I think the project has fulfilled its main goal and I was able to develop new enjoyable mechanics for this motion game. Also I widely developed my javascript skills and I got very familiar with the Phaser engine. Besides that this was a great experience to work on a real project and on a product that will be on the market soon, therefore it helped me to feel what is it to be on a real working environment and learn a lot about real work strategies and teamwork.