Game Design Prototype Development

This paper describes and shows how the game design prototype was developed, how developed elements introduce interesting choices, affects dynamic and aesthetics, and how new elements change the gameplay. Moreover, paper presents a list of changes made according to feedback provided by other developers. Last, paper provides description of critiqued new elements made by other developers and suggestions for these elements' improvement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AoE	Area of Effect
DPS	Damage per second
HP	Health points
PvC	Player versus Computer
PvP	Player versus Player
TD	Tower Defense

Process Description and Development

Before starting to implement different gameplay elements, author have analysed the game mechanics and created "gameplay" loop (see Figure 1) to understand what elements in majority forms the user experience (Bycer, 2019).

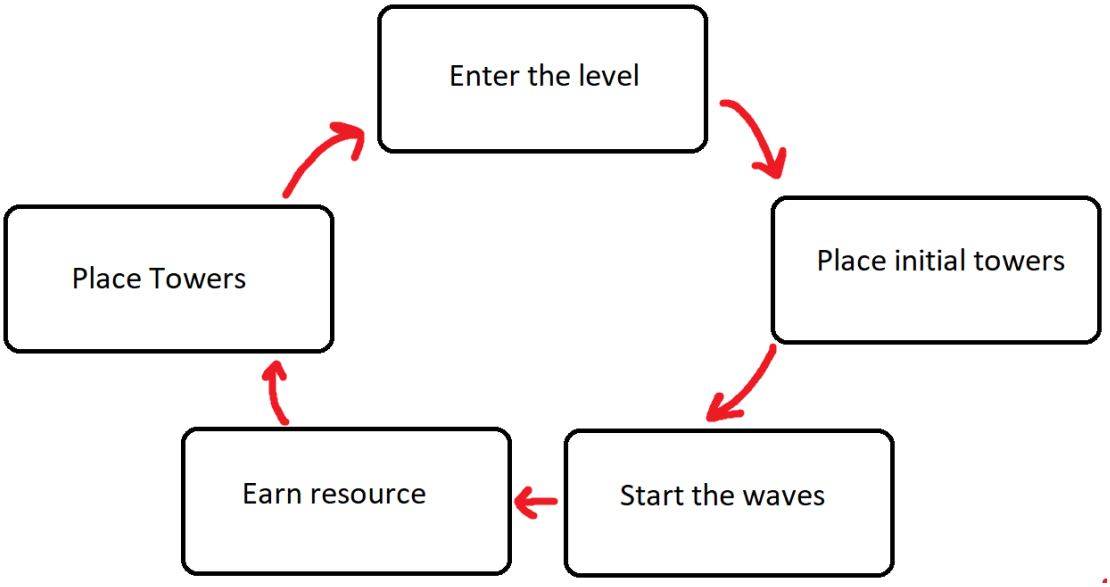

Figure 1 Gameplay core mechanics

Tower Element Development

Conceptualizing and Designing

After own playtesting and observing other people playtest existing prototype author felt that support towers are not being used often enough. Based on that author decided to create another support tower.

Initial idea was to create a tower which may improve or worsen the game situation for player depending on random. But as randomness may limit ability to see the result of player game knowledge and skill what leads to decreasing fun (Holopainen, 2021) – it was decided to change the mechanic of tower, but saving it support property with obligated tactical placement. As the result, tower named "Rage Commander" which increases fire rate of towers and enemy speed was created.

Implementation

Firstly, by following documentation available, author was able to create a simple tower which does nothing (see Figure 2).

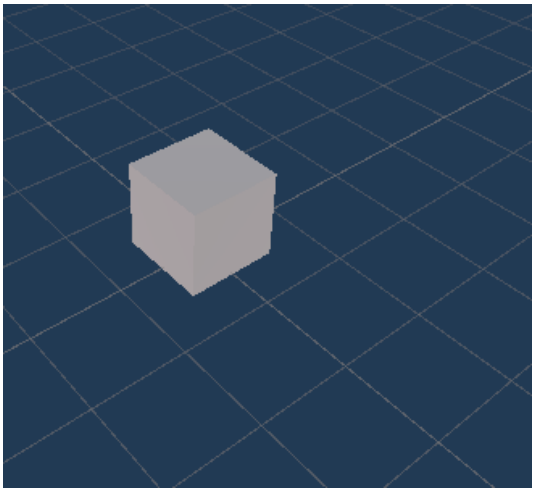

Figure 2 New tower model

To create AoE support tower, firstly author decided to investigate what scripts (see Figure 3) of EMP tower are responsible for slowing enemies down and copy them to new tower created.

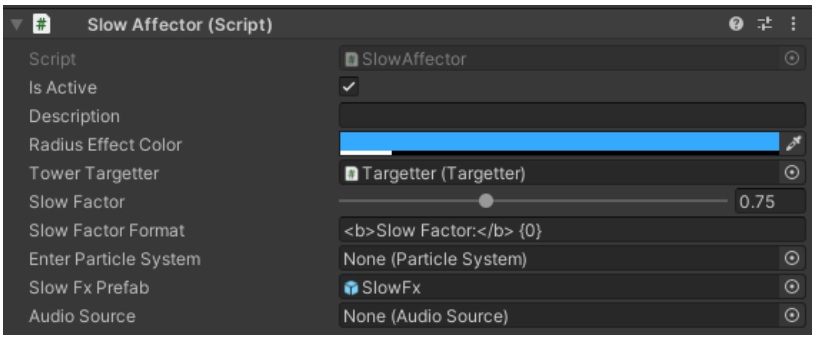

Figure 3 Script responsible for slowing enemies down

Afterwards, author checked how affector works by viewing a code and reading official documentation (see Figure 4).

```
/// <summary> Attaches a AgentSlower to the agent
1 reference
protected void AttachSlowComponent(Agent target)...

/// <summary> Removes the AgentSlower from the agent once it leaves the area
1 reference
protected void RemoveSlowComponent(Agent target)...

/// <summary> Fired when the targetter aquires a new targetable
2 references
protected void OnTargetEntersRange(Targetable other)...

/// <summary> Fired when the targetter aquires loses a targetable
2 references
protected void OnTargetExitsRange(Targetable other)...
```

Figure 4 Slow affector code

Then, author investigated how tower triggers only on enemies, but not on towers. The “teams” in provided prototype are represented as alignments. So, author created new one called “Support Tower” which have “Player” (towers) and “Enemy” alignments as opponents to trigger on them (see Figure 5).

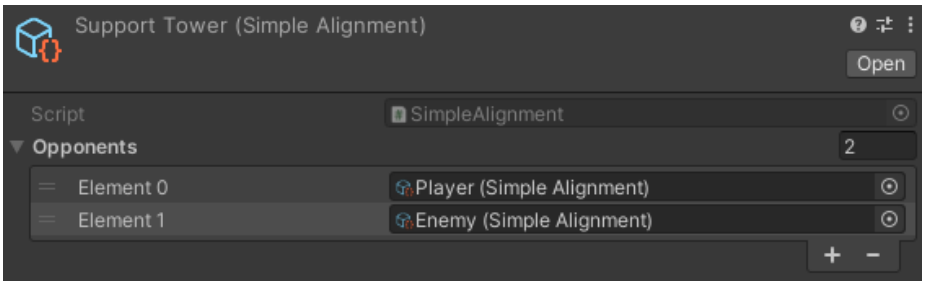

Figure 5 "Support Tower" alignment

After creating custom alignment, author copied SlowAffector script and adopted it for fire rate component append for towers (see Figure 6).

```
protected void AttachFireRateComponent(GameObject target)
{
    // Add Agent
    var agent = target.GetComponent<AgentFireRateUp>();
    if (agent == null)
    {
        agent = target.gameObject.AddComponent<AgentFireRateUp>();
    }

    if (fireRateFactor >= 1)
    {
        agent.Initialize(fireRateFactor, fireRateFxPrefab);
    }
    else
    {
        agent.Initialize(fireRateFactor, fireRateFxPrefab, position: default(Vector3), scale: 1, positiveMode: false);
    }


    // SFX
    if (enterParticleSystem != null)
    {
        enterParticleSystem.Play();
    }
    if (audioSource != null)
    {
        audioSource.Play();
    }
}
```

Figure 6 Attach fire rate component function

As SlowAffector is responsible for attaching slow component (AgentSlower) to any target which enters the range, new component based on it called AgentFireRateUp was created. Moreover, AgentSlower was edited so that is might make enemies faster (see Figure 7).

```
// Slow mode or speed up
if (positiveMode)
{
    // find greatest slow effect
    float min = slowFactor;
    foreach (float item in m_CurrentEffects)
    {
        min = Mathf.Min(min, item);
    }
    newSpeed = originalSpeed * min;
}
else
{
    // find greatest speed up effect
    float max = slowFactor;
    foreach (float item in m_CurrentEffects)
    {
        max = Mathf.Max(max, item);
    }
    newSpeed = originalSpeed * max;
}
```

Figure 7 Negative slow factor

As towers class varies from enemy by behaviour and logic, the author has changed some fields and fire rate affector applying procedure (see Figure 8).

```
// Behaviour.
affector = m_Tower.gameObject.GetComponentInChildren<AttackAffector>();
if (affector == null) return;

// Save origin fire rate only once.
if (originalFireRate == -1)
{
    originalFireRate = affector.fireRate;
}

float newFireRate = 0;

// Fire rate up or down
if (positiveMode)
{
    // Fire rate up
    float max = fireRateFactor;
    foreach (float item in m_CurrentEffects)
    {
        max = Mathf.Max(a: max, b: item);
    }
    newFireRate = originalFireRate * max;
}
else
{
    // Fire rate down
    float min = fireRateFactor;
    foreach (float item in m_CurrentEffects)
    {
        min = Mathf.Min(a: min, b: item);
    }
    newFireRate = originalFireRate * min;
}
```

Figure 8 Code snippet responsible for applying new fire rate

As in prototype initially had no option of reapplying affector on leveled up tower or enemy, author have created such from scratch by observing different scripts and inserting them in Update method (see Figure 9) as action assigning based on tower upgrade did not work as expected (see Figure 10).

```
private void Update()
{
    try
    {
        if (affector == null) affector = m_Tower.gameObject.GetComponentInChildren<AttackAffector>();

        if (oldFireRate != affector.fireRate)
        {
            originalFireRate = affector.fireRate;

            // Fire rate up
            float max = fireRateFactor;
            foreach (float item in m_CurrentEffects)
            {
                max = Mathf.Max(a: max, b: item);
            }
            affector.fireRate = originalFireRate * max;
            oldFireRate = affector.fireRate;

        }
    }
    catch (NullReferenceException)
    {
        //
    }
}
```

Figure 9 Update affector behaviour for leveled up tower

```
m_Tower.towerUpgrade += () => UpdateAgent();
```

Figure 10 Action based function call

In total new tower was created (see Figure 11) and placed in level grid (see Figure 12).

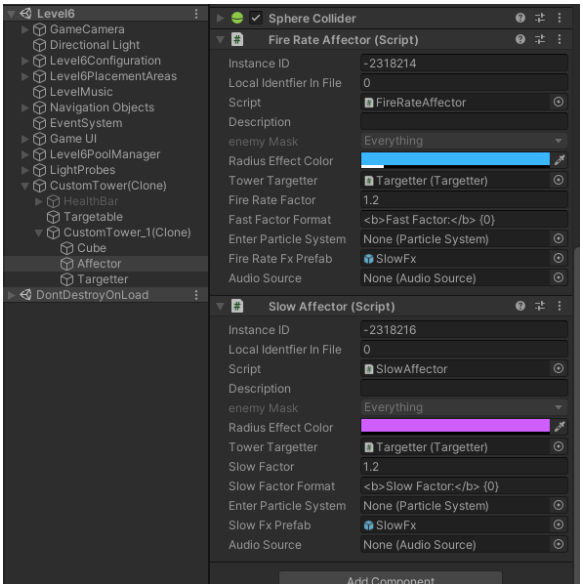

Figure 11 New tower in the inspector

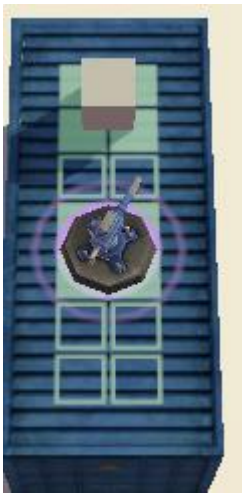

Figure 12 New tower in the game

Balancing

To balance the tower, author have planned new tower price based (see Table 1) on risk-reward system (David et al.,2011). Selling price was calculated based on other prototype provided towers.

Table 1 New tower buy/sell amount

Level	Price, resource units	
	Buy	Sell
1	10	5
2	15	12
3	25	22

Last, to balance tower from gameplay point of view, author have created multiple levels with different percentage of fire rate and speed up enemies increase (see Table 2).

Table 2 New tower effects description

Name of effect	Level of tower		
	1	2	3
Tower Fire Rate	+20%	+35%	+50%
Enemies Speed up	+20%	+30%	+40%

Introduced mechanics and gameplay change

New tower introduces “interesting” or “meaningful” choice (Schell, 2015, 210-214) to have or not possibility to increment fire rate for towers and speed for enemies. Also, by introducing different tower levels with different effect ratio (see Table 3), author stimulates user to upgrade introduced tower to highest level possible to get maximum efficiency from it.

Table 3 New tower effect ratio according to tower level

Level of Tower	Effect Ratio (fire rate: speed up), %
1	50:50
2	53.85:46.15
3	55.5(5):44.4(4)

Dynamics as pairing with different type of towers arise. Moreover, dynamic of strategic placement of new tower appears – user need to place tower so it will only increment fire rate for towers, but not increment enemies speed (see Figure 13).

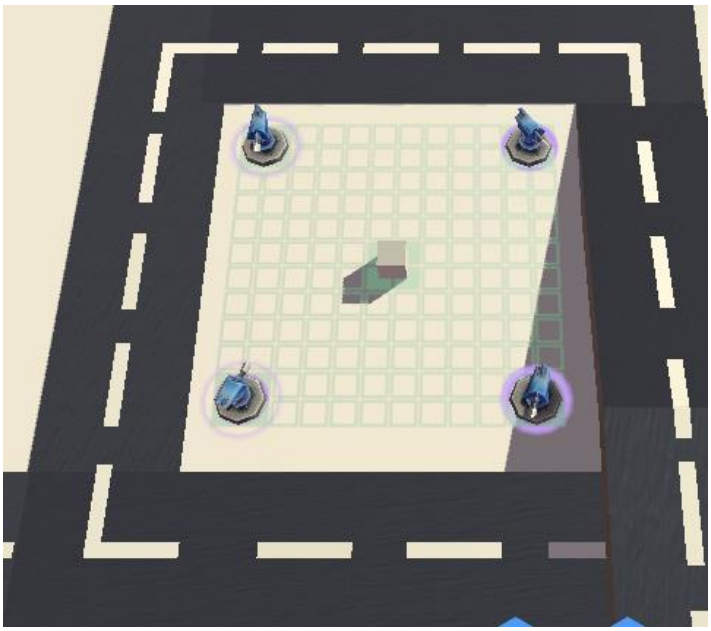

Figure 13 Variant of placement of new tower

By combining all previously mentioned dynamics there are few aesthetics arise. Main one is challenge - to beat the newly created level, user is expected to master new tower, which they did not used previously. After adding a tower, an expression aesthetics also arise - now player can be more creative by experimenting placing newly created tower in pair with others.

Enemy Element Development

Conceptualizing and Designing

As support tower was created and in prototype given there is a lack of any support type enemies, initial concept of enemy was also to make support type enemy. To find interesting enemy mechanics which might be adopted author overviewed such games as “Orcs Must Die! 2” (Robot Entertainment, 2012), “Kingdom Rush Frontiers” (Ironhide Game Studio, 2013) , “Kingdom Rush Vengeance” (Ironhide Game Studio, 2018) etc. In most of these games, support type enemies heal teammates or revive dead ones, gives resist versus special tower type, or summons new enemies.

The author does not like any of previously mentioned ideas and came up with breaking economy-based enemy. Every tower defence game has economy based on resource (in our prototype energy), but as in real life, resource is limited. That is why players tries to maximize efficiency (in our case DPS) using least resources possible. To make player think about economy and future risks author created new enemy type called “Energy Goblin”.

Implementation

By following documentation guide, new enemy type which trying to get to base was created (see Figure 14).

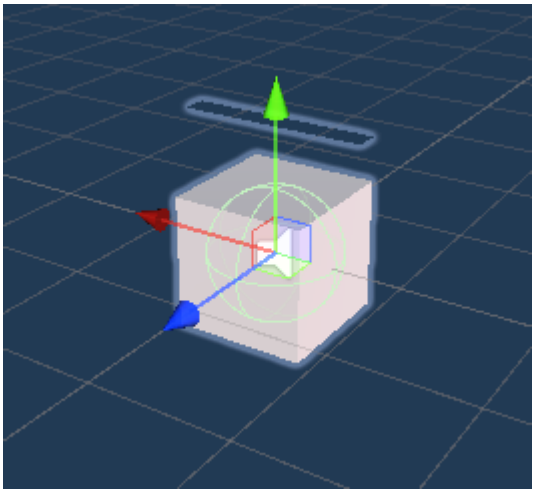

Figure 14 "Energy Goblin" enemy type

After new enemy was created author have checked how LootDrop script is working to understand how balance system is built (see Figure 15).

```
protected virtual void OnDeath(HealthChangeInfo info)
{
    m_DamageableBehaviour.configuration.died -= OnDeath;

    if (info.damageAlignment == null ||
        !info.damageAlignment.CanHarm(m_DamageableBehaviour.configuration.alignmentProvider))
    {
        return;
    }

    LevelManager levelManager = LevelManager.instance;
    if (levelManager == null)
    {
        return;
    }
    levelManager.currency.AddCurrency(lootDropped);
}
```

Figure 15 LootDrop script balance system interaction

Afterwards, author have created new script named StealMoney (see Figure 16) and assigned it to new enemy game object.

```
public class StealMoney : MonoBehaviour
{
    private LevelManager levelManager;
    public int currencyToSteal = 10;

    Unity Message | 0 references
    void Start()
    {
        levelManager = LevelManager.instance;
        levelManager.currency.AddCurrency(-currencyToSteal);
    }
}
```

Figure 16 StealMoney script

Script Start method is being called when “Goblin” is spawned and takes 10 energy from user and after death returns it because of Loot Dropped value in LootDrop scripts (see Figure 17).

Figure 17 Loot Dropped and Steal preview

Example of active “Goblin” mechanic can be seen in Figure 18.

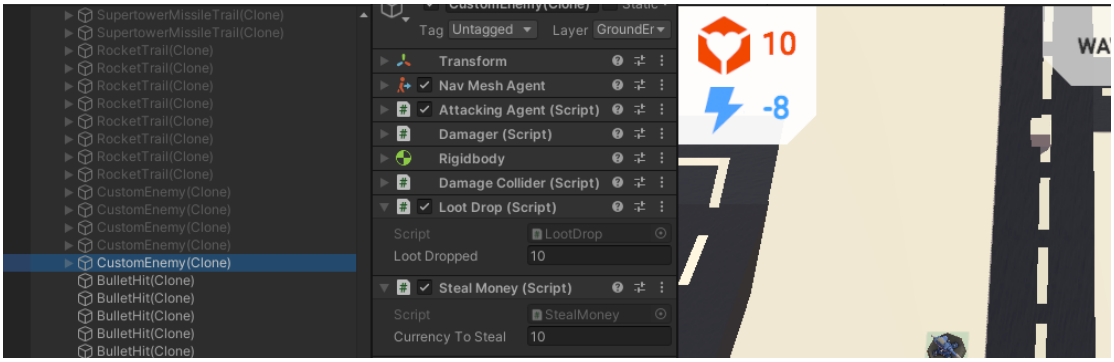

Figure 18 "Energy Goblin" mechanic example

Introduced mechanics and gameplay change

Such enemy mechanic is to steal 10 energy from user and after being killed return all the stolen energy. “Interesting” choice from such mechanics arise - user can choose between saving money for “unlucky” moment or risk and buy a tower on last money and possibly get resources in negative amount.

Also, by introducing such mechanic new dynamic appears - new enemy spawn might ruin plan to upgrade or buy a tower. Another one is limited time to place towers, as user may never know exact timings when such enemy will spawn.

All these dynamics leads to challenge aesthetics, where player should spend resources neat and accurate to have stable economy and pass the level perfectly.

Level Design Development

Conceptualizing and Designing

After reviewing existing levels in different TD games such as Kingdom Rush (Ironhide Game Studio, 2011), GemCraft - Frostborn Wrath (Game in a Bottle, 2020), Dungeon Defenders II (Trendy Entertainment, 2017) etc., author tried to develop own level with unique interactive gameplay mechanic.

Firstly, author placed a start and end in level, after that tried to develop “interesting” path. There were some variants (see Figure 19) which were rejected because of complexity of path enemy to go.

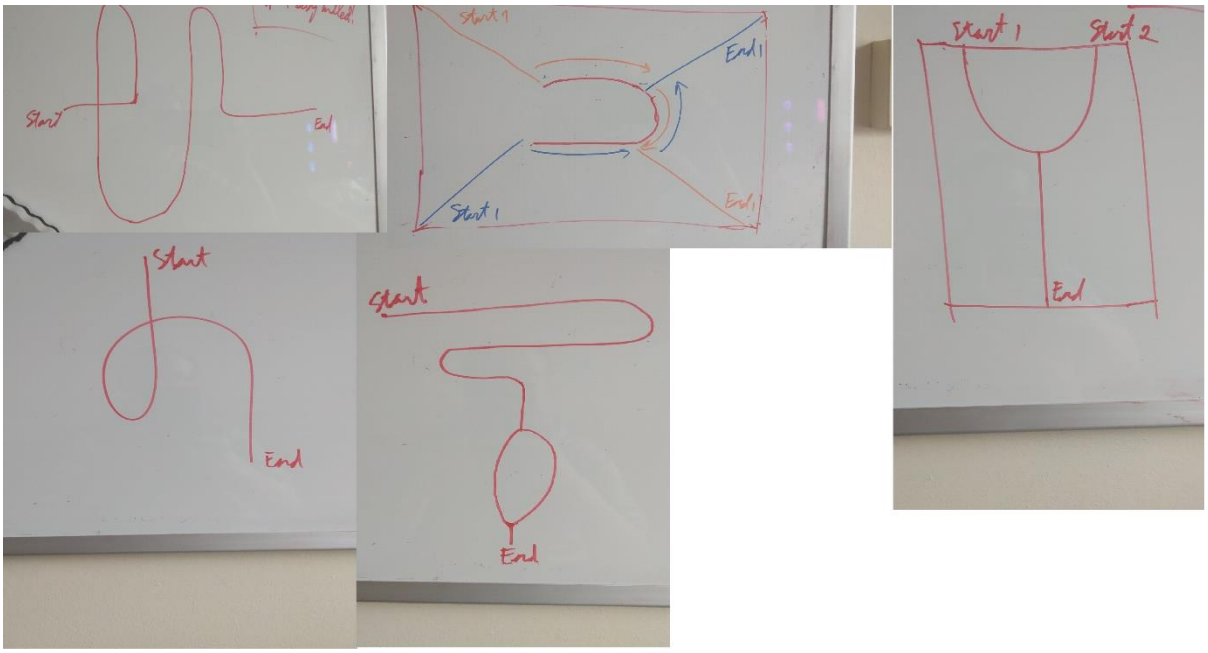

Figure 19 Initial concepts

After a while, author found that most of the levels in TD games have same level design pattern. Each level has blind roads or spots where enemies can go through and do not get any damage (or get minimal amount), what is being achieved by removing spots for placing towers. In the same time there are loops where player can place a lot of towers and maximize total damage done.

In some levels, usually in the end, there is “split road” element, there is average introduced 1 challenge aesthetic - difficulty to control all the paths (in the beginning of the game). Moreover introduces 1 dynamic - splitting strong enemies’ groups (presume weak are being destroyed number of spots for towers comparing to different level parts).

Implementation

After such analysis, author merged all these elements and created a level concept (see Figure 20).

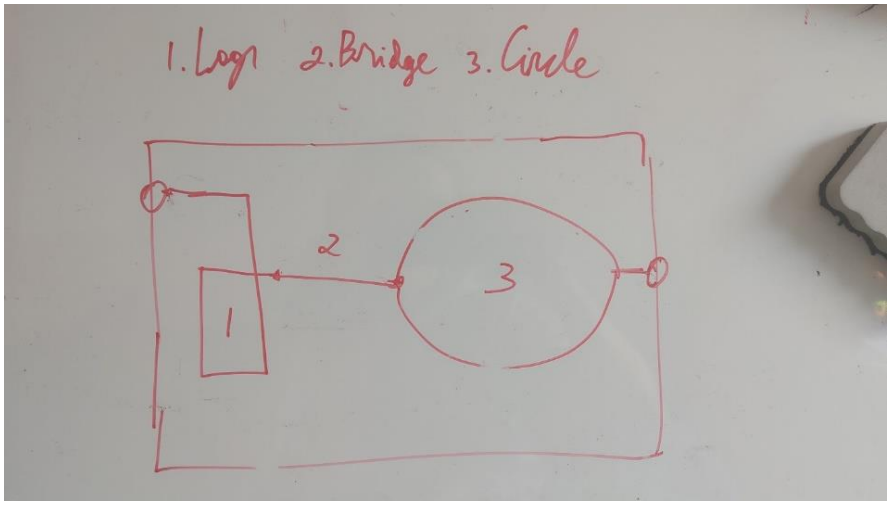

Figure 20 Accepted level concept

Using iterative game design approach (which is described further) author created initial prototype with path to go and spots to place towers (see Figure 21).

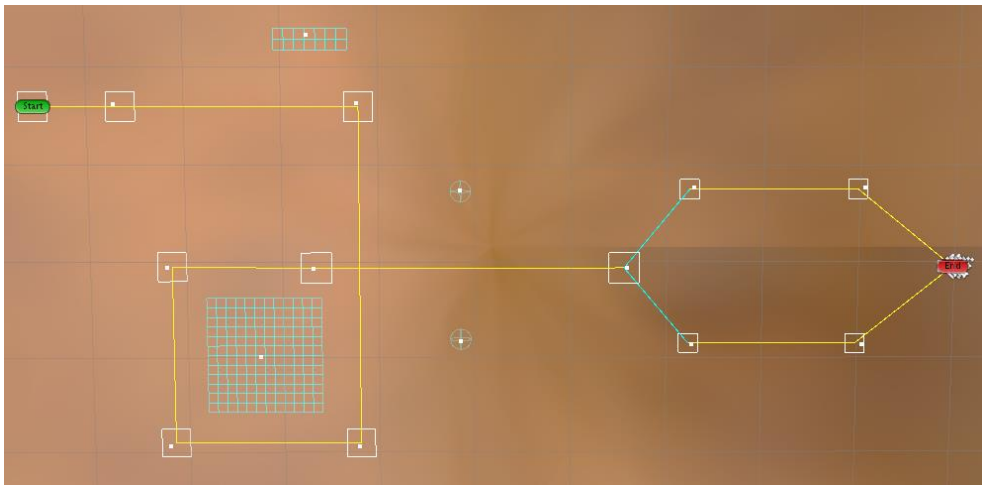

Figure 21 First iteration of level prototype

After first iteration of play testing and evaluation process, author edited level layout to adjust difficulty by creating new road called “Loop Start” where enemies can go from only starting by the end of 2nd wave. Moreover, by adding another road called “Start2” author make harder to predict when and which enemies will form a group because of different length of initial roads and different speed of enemy agents (see Figure 22).

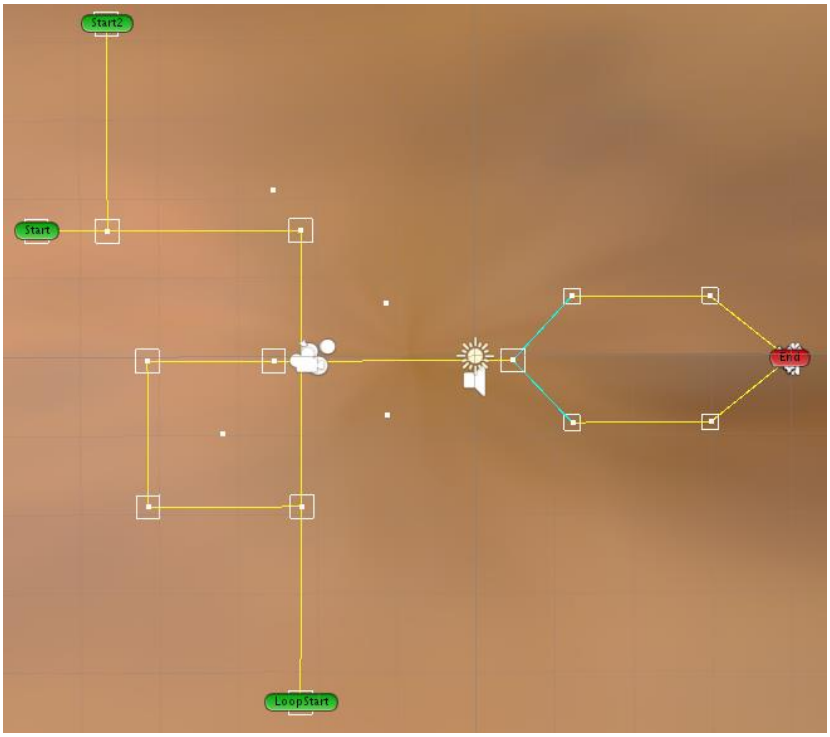

Figure 22 Second iteration of level prototype

First is entry to the loop and loop itself, where player can place a lot of different towers to defeat most of enemies. Optimal towers in this part are with high fire rate (as Assault Cannons), AoE damage (Rocket Platforms) and support towers as EMP Generator or Rage Commander which will slow down enemies or increase fire rate accordingly (see Figure 23 and Figure 24).

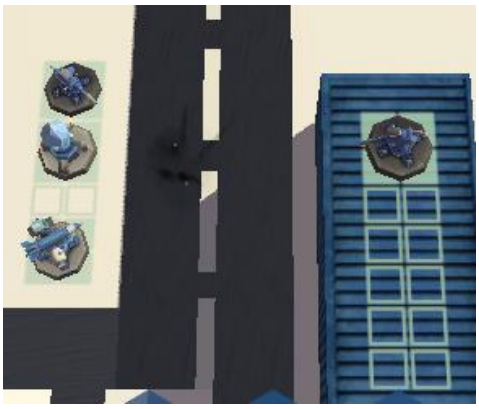

Figure 23 Entry to the loop optimal placement

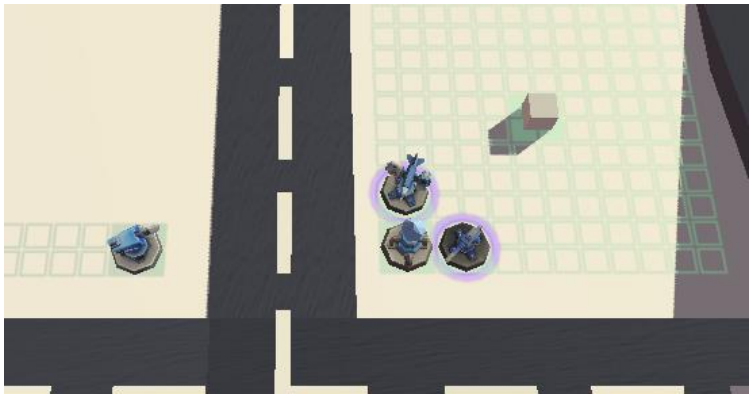

Figure 24 Optimal tower placement in loop

Then goes “bridge” element, chokepoint, where are only few spots for towers. Optimal are Plasma Lance towers as they have huge range of attack (might attack enemies going through “Loop” level part), high damage, but small fire rate (see Figure 25).

Figure 25 Optimal "bridge" part configuration

Last, goes “split” or “circle” element where any towers can be placed in average amount. As this part of the level is last, optimal strategy may vary from game situation, but author would recommend Missile Array, Rocket Platforms and Assault Cannons (see Figure 26).

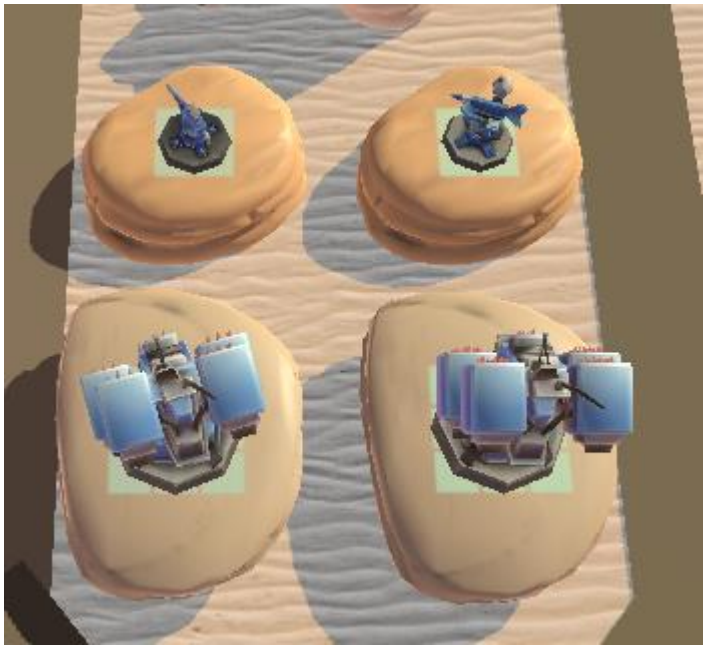

Figure 26 Desert optimal strategy

Balancing

To make possible to beat the level and balance of resources and difficulty, author created custom wave configuration which was few times changed according to further feedback given. Each wave in total was configured so that player have economy freedom in choosing towers, but still should think about its' tactical placement (see Table 4).

Table 4 Resource gain during Level 6 waves

Number of Wave	Total Resource Gain
0 (Pre-game)	30
1	18
2	43
3	60
4	30
5	42
6	42
7	82

Waves enemy types were configured (see Table 5) using flow theory (Csikszentmihalyi, 1990), where initially difficulty is low and by placing only few towers player can earn a resource, not feeling anxiety yet. After that difficulty increases, but not for much to restrain player from frustration. On 3rd wave difficulty increased once more and this time reaches peak value comparing to previous waves. After completing hard wave, player is being rewarded with easy one, where player can gain more resources for completing further waves. On 5th wave difficulty

increases again and is approximately on the same level as on 2nd wave due to player progression and no “Energy Goblin” enemy spawn. Afterwards, 6th wave is like previous, but has Energy Goblins which may cause inconvenience for the player. Last, 7th wave is most complex wave, where difficulty reaches its peak for all the time. Each type of enemies is being used; Super Boss is summoned in the end representing the final challenge for the player.

Table 5 Enemy types spawn during Level 6 waves

Number of Wave	Enemy types amount								
	Buggy	Copter	Tank	Boss	Super Buggy	Super Copter	Super Tank	Super Boss	Energy Goblin
1	18	0	0	0	0	0	0	0	2
2	13	6	2	0	2	0	0	0	3
3	12	2	5	0	7	2	0	0	3
4	6	6	1	0	1	0	0	0	2
5	2	2	1	1	2	3	1	0	0
6	2	0	1	1	7	2	1	0	2
7	7	1	1	2	8	2	2	1	3

Introduced mechanics and gameplay change

As new tower is also introduced on this level, level tests not only mastery of placing previous towers, but also requires using new one for successful completing the level. As user does not have any experience with it, author have created place where user can achieve maximum efficiency of new tower and minimizing the risks (see Figure 27). After user will master using the tower mechanic, they may experiment with riskier tower placement as placing in pair with other towers.

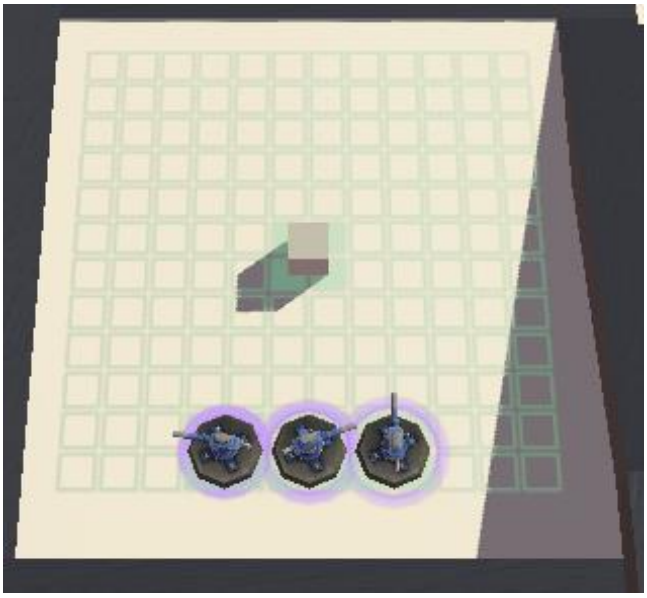

Figure 27 Optimal place for new tower

To ensure synergy between level and new enemy, author added new enemy almost in each wave. To make sure user understands how to control it, author configured new enemy spawn time (see Figure 28) so that in the beginning, spawn of new enemy is not so critical, and player may even miss it as they usually spawns in the end of the wave and does not really affect the wave gameplay.

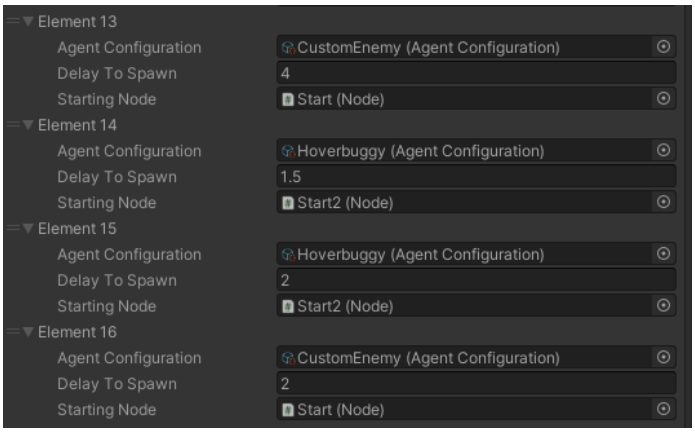

Figure 28 1st wave new enemy spawn time configuration

With increasing difficulty of waves, new enemy spawns in more inconvenient situations for the user as in pair with more strong enemies (see Figure 29) or in pack of few weak ones.

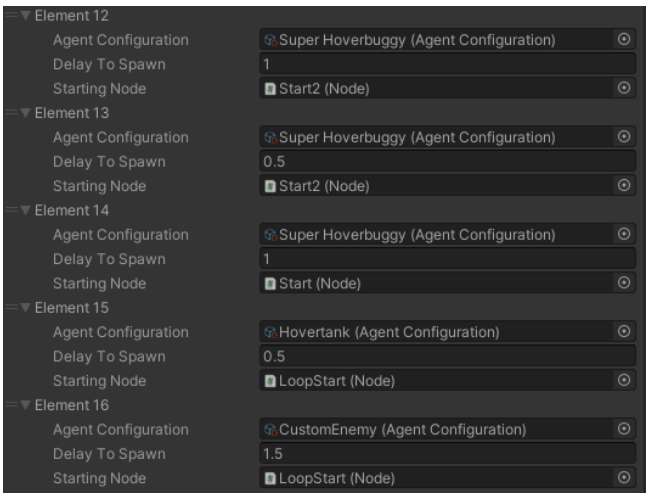

Figure 29 3rd wave new enemy spawn time configuration

In total by combining new level with dynamics of new enemy and new tower, aesthetics alliance of challenge and sensation appears.

Challenge aesthetics is based on economics planning and strategic development involving new tower placement and using multiple enemy roads.

To ensure player gets sensation aesthetics author created context for the level (see Figure 30 and Figure 31) using “visual” language (Taylor, 2013). Context of the level is following - left part of it is a town, where some containers and other city elements are being placed. After that, the “bridge” element represents a bridge itself. To strengthen sensation aesthetics even more author created dynamically changing over time river by using shader graph. Right and last part of the level is desert, where some rocks and cactuses present.

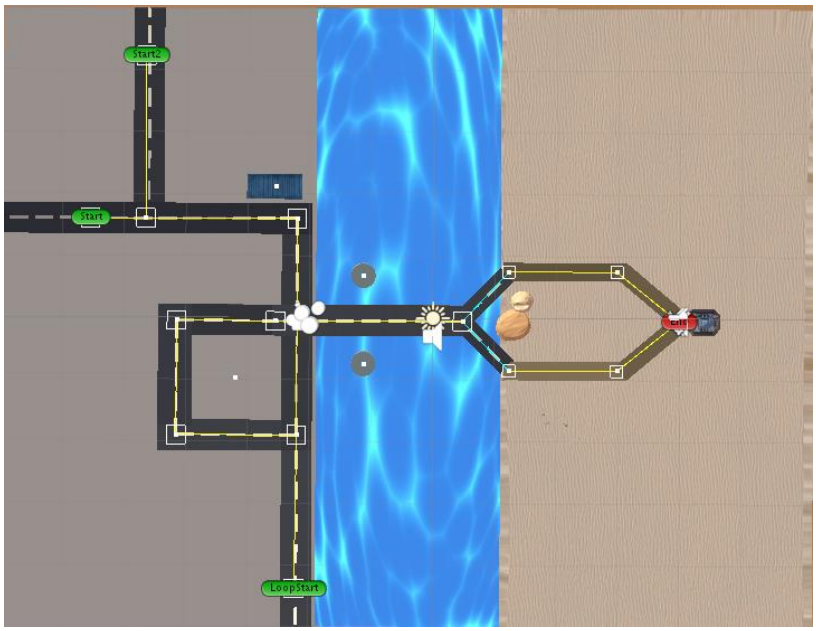

Figure 30 Creating context for level prototype

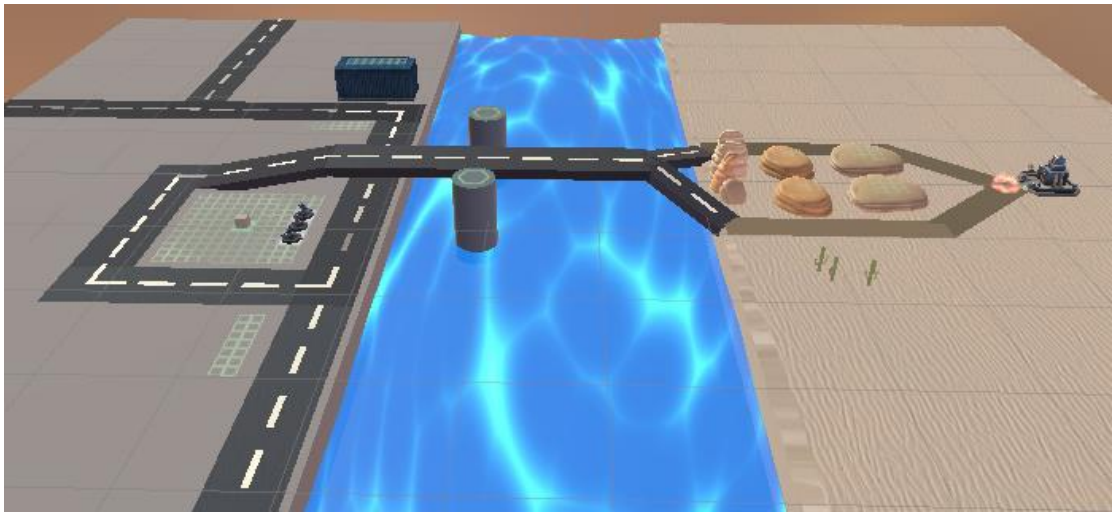

Figure 31 Angled view of created level

Theory

MDA

MDA framework (Hunicke R. et al., 2004) is a tool for analysing the game by identifying programmed mechanics, dynamics which consist of few mechanics and finally aesthetics consisted of few dynamics (see Figure 32). MDA analysis was used in this project for description of new elements as well as clarification for changes provided (which are discussed further).

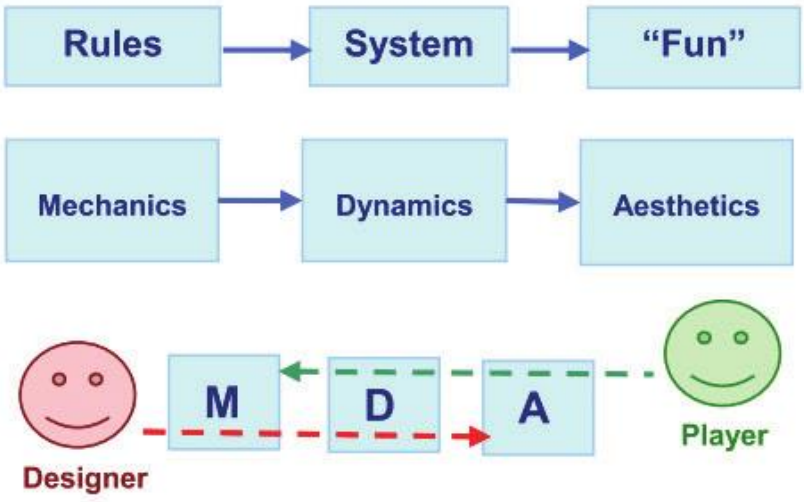

Figure 32 MDA diagram (Hunicke R. et al., 2004)

Iterative Model Design

In level design process author have used iterative model instead (Macklin and Sharp, 2016) of another commonly used - waterfall one (Royce, 1970). In iterative model the workflow has unique structure way (see Figure 33).

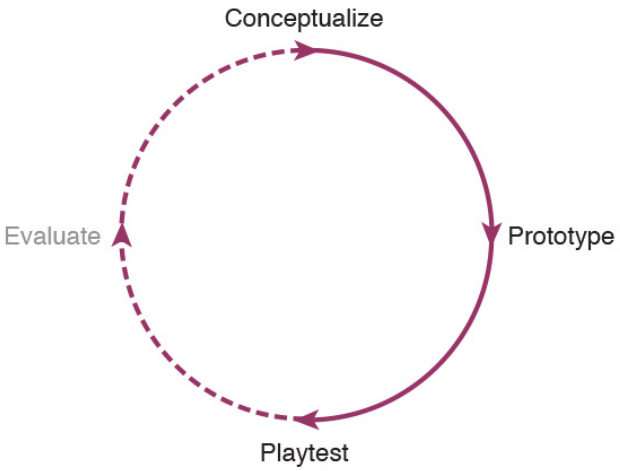

Figure 33 Iterative design model (Macklin and Sharp, 2016)

Firstly, comes development of the concept based on generation of ideas, where details are not important. Afterwards comes prototyping when concept is being transferred from ideas into the game. Then, comes playtesting process step, which reveals what is or isn't working in developed prototype. Last, comes evaluation process, where author decided what to change in existing prototype depending on feedback provided.

Flow Theory

Moreover, to improve the gameplay of the level and game overall, macro flow principle was followed (see Figure 34). Flow theory is mostly about users' skills, progression in the game and their engagement. If user plays a game and find that easy, they will feel boredom, otherwise if game is hard – anxiety. The key is to balance between these phases and reach engagement feeling, what was achieved by configuring custom difficulty waves.

Figure 34 Macro flow diagram (VOiD1 Gaming, 2020)

Core Gameplay Loops

Another theory piece is core gameplay loops (Bycer, 2019). The idea beneath this theory is structuring some mechanics in circle way to understand what exactly forms most of playtime experience. In this prototype gameplay loops helped author to understand what core mechanics are and what might be changed or improved by the author by introducing new tower and enemy types.

Interesting choices

The idea of any interesting choice (Alexander, 2012) is that each user decision has consequences. Each decision should have a balance of risk and reward. Author achieved such choices by introducing new enemy (economics related choice) and new tower (tactical position related choice) by this making prototype even more interesting to play.

Fairness

Fairness might be presented as equal probability of making a mistake (Iida, 2007). By trying to balance the game, author tried to achieve fairness and depth, because only based on these factors genuine balance appears (Tynan, 2013). By fairness author introduces tactical balance such as tower placements grids and custom waves. For achieving depth balance author have balance strategies among which the player chooses in any given situation.

Feedback and Changes

In total author got 4th feedbacks from following developers:

- Thomas Wilcockson (25273418@ students.lincoln.ac.uk)
- Finlay Robb (25195541@students.lincoln.ac.uk)
- Raine Reynolds-Osborne (25151585@stunents.lincoln.ac.uk)
- Sophie Chau (25054451@stunents.lincoln.ac.uk)

Each of the reports was made in different time and each developer had different versions of the game to test. Also, each report identified at least some improvement may be done to improve the game as well as some positive feedback about it. After each report the author edited the prototype and send to next developer.

In the first report, the author of report pointed that he liked the idea of strategic based tower (see Figure 35) and it should cost a little bit more.

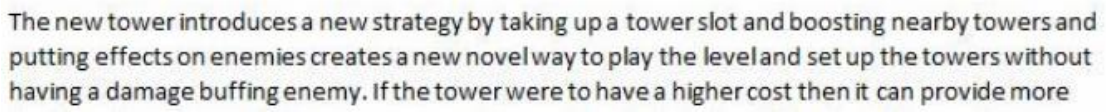

Figure 35 Good idea, increase cost of the tower

Moreover, the author pointed that new enemy should spawn more often to create a greater risk (see Figure 36). In the same time found that it is hard to get past wave 4 without using cheats. Furthermore, author proposed to add more grids in the beginning of the level and remove some from the end of the level.

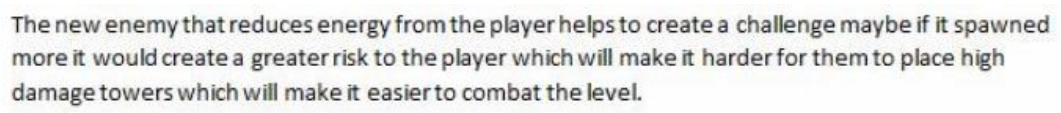

Figure 36 More new enemy entities needed

Next report author suggested adding text about enemy in text description and audio or visuals to mark enemy have stolen the energy. Also, author pointed out that camera feels too zoomed in (see Figure 37).

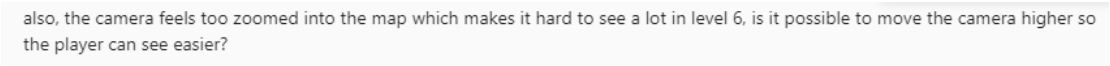

Figure 37 Fix camera zoom

Third report showed that enemy is unbalanced, and energy taken from player should be decreased (see Figure 38).

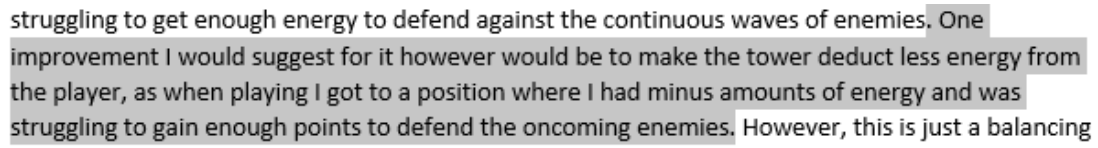

Figure 38 Enemy is unbalanced issue

Last, fourth report pointed out that level does not have enough grids in the start and end of the level (see Figure 39).

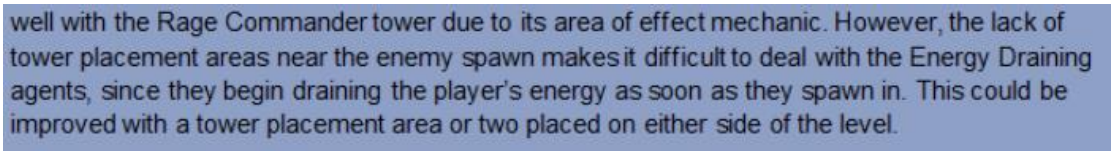

Figure 39 Not enough grids provided

After getting a report the author has analysed what can be improved and why it should or should not. In total following items being fixed after all the reviews:

- Tower buy price was reduced from 15 to 10 to make tower more gameplay balanced (Figure 40).

Figure 40 Decreased tower price

- Custom waves were configured for improving flow feeling
- Some tower grids were moved and added in the beginning for gameplay balance (see Figure 41)

Figure 41 Moved and added grids for towers

- Distanced the camera from level to increase sensation aesthetics
- Edited amount energy deducted from 10 to 7 to make new enemy more balanced and fairer (Iida, 2007) for player (see Figure 42)

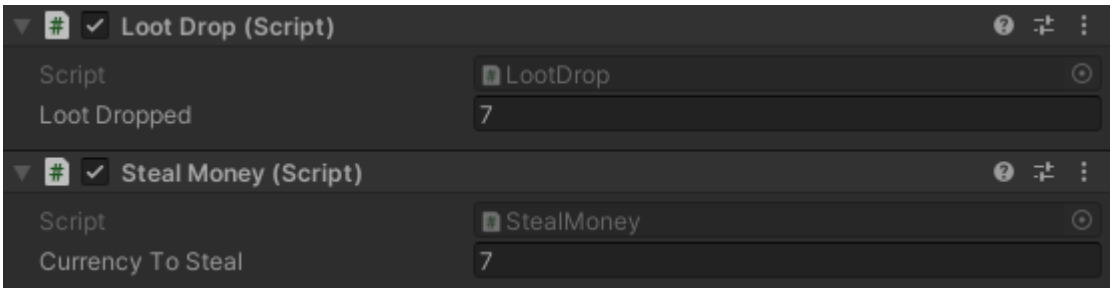

Figure 42 Reduced energy deducted from user

Critique and Improvements

Feedback was provided to Sophie Chau (25054451@students.lincoln.ac.uk).

Enemy

This developer introduced new type of enemy – healer, which heals the enemies located close to him (see Figure 43). The enemy felt well balanced and its spawns with appropriate timing. In total no possible suggestion for improvement were made.

Figure 43 New enemy (Chau, 2021)

Tower

New tower (see Figure 44) brings unique type of gameplay – poisoning enemies. The problem with it is that by placing a lot of such towers, their poison effect stacks up and deal incredible amount of damage. Also, damage dealt by tower for its' price (8) felt too high. In total author suggested applying only 1 poison affector at time but increase poison damage to save balance of the tower and make placement of new tower more strategically responsible.

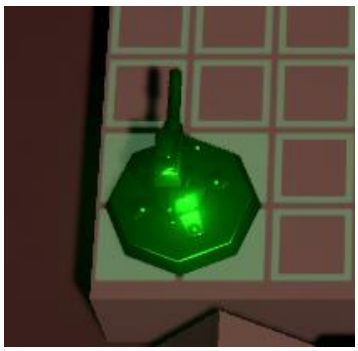

Figure 44 New tower (Chau, 2021)

Level

The first impression from introduced level was it is hard to navigate, as there was no base objects and no "visual language" as lights, geometry, colour and animation (Taylor, 2013) to help player to understand where enemies will come from (see Figure 45) and to (see Figure 46).

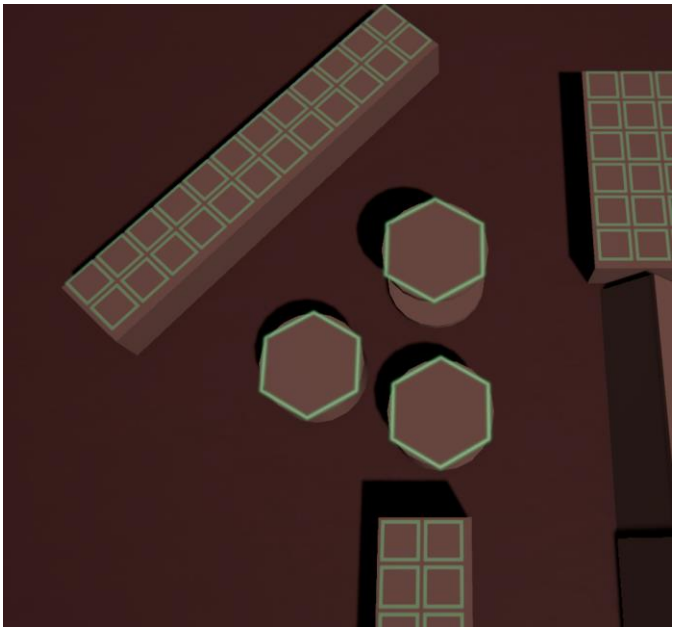

Figure 45 Start of the level (Chau, 2021)

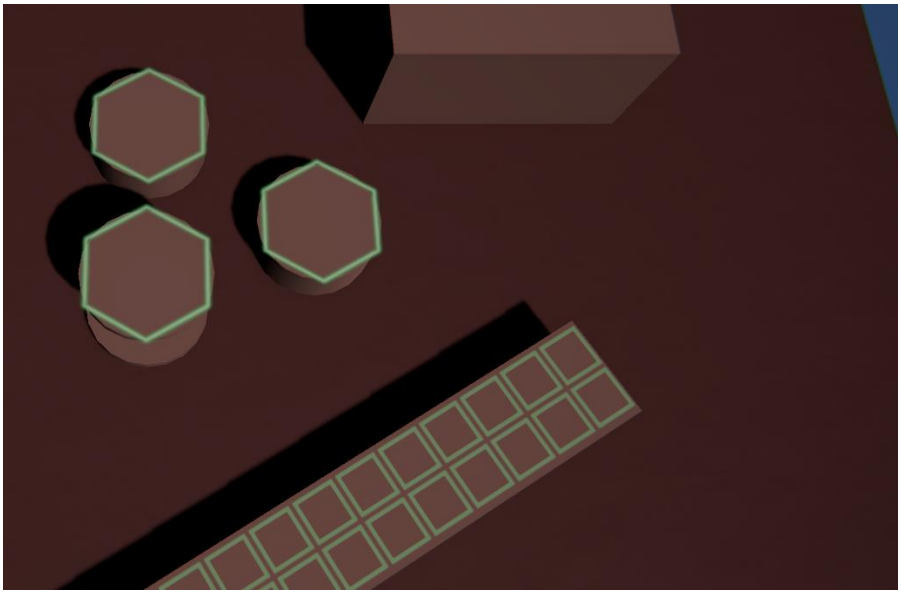

Figure 46 End of the level (Chau, 2021)

Also, newly introduced level had a lot of grids for placing towers in middle and start stages of level (see Figure 47), author suggested to remove 20-30% from provided amount.

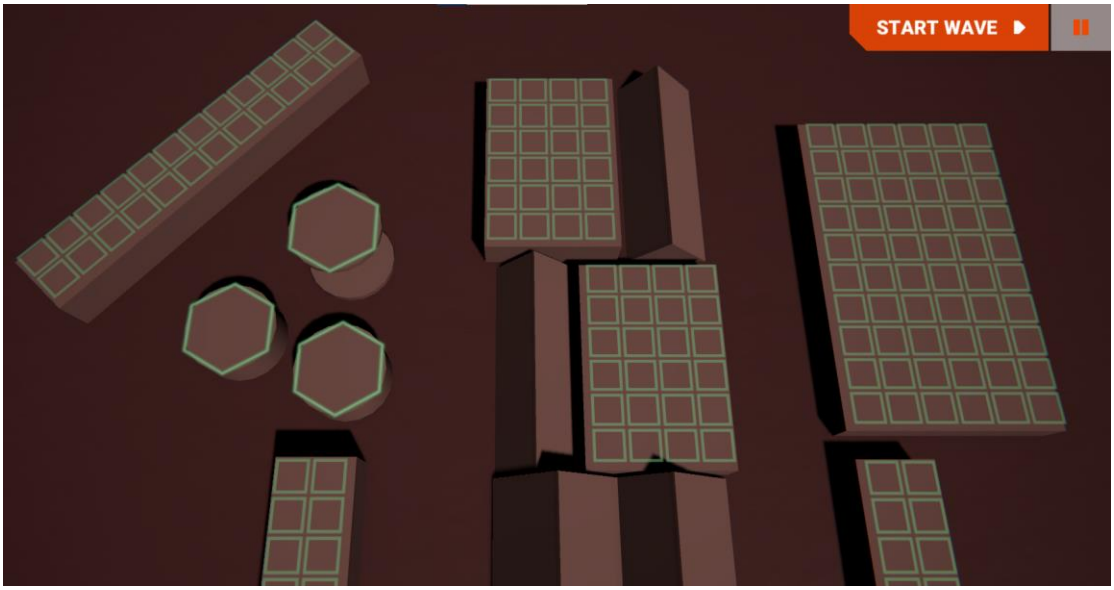

Figure 47 Start and centre grids for placing towers (Chau, 2021)

Result

Summarizing, this paper described new element development workflow and how each from introduced elements affect the gameplay from various points of theory game design.

Moreover, theory such as: MDA framework, iterative design approach, game loops, flow theory and interesting choices were discussed.

Last, this paper provides a list of edits made because of got feedback as well as describes what type of critique was sent to other developers.

References

Alexander, L. (2012) *GDC 2012: Sid Meier on how to see games as sets of interesting decisions.* Available from: https://www.gamasutra.com/view/news/164869/GDC_2012_Sid_Meier_on_how_to_see_games_as_sets_of_interesting_decisions.php [Accessed 13 May 2021].

Bycer, J. (2019) *Why the Core Gameplay Loop is Critical For Game Design.* Available from: https://www.gamasutra.com/blogs/JoshBycer/20190425/341208/Why_the_Core_Gameplay_Loop_is_Critical_For_Game_Design.php [Accessed 14 May 2021].

Csikszentmihalyi, M. (1990) *Flow: The psychology of optimal experience* (Vol. 1990). New York: Harper & Row. Available from: https://www.researchgate.net/publication/224927532_Flow_The_Psychology_of_Optimal_Experience [Accessed 11 May 2021].

Chau S. (2021) *Tower Defence Template* [game].

David, E., Paul, W., Nesbitt Keith, V. and Ami, E. (2011) *Balancing Risk and Reward to Develop an Optimal Hot-Hand Game*. 11(1). Available from http://gamestudies.org/1101/articles/williams_nesbitt_eidels_elliott [Accessed 18 May 2021].

Game in a Bottle (2020) *GemCraft - Frostborn Wrath* [game]. Available from: https://store.steampowered.com/app/1106530/GemCraft__Frostborn_Wrath/ [Accessed 18 May 2021].

Holopainen J. (2021) Chance and Randomness [PowerPoint presentation]. *CGP1008M: Game Design.* Available from: https://blackboard.lincoln.ac.uk/webapps/blackboard/content/listContent.jsp?course_id=_146482_1&content_id=_4094740_1 [Accessed 10 May 2021].

Hunicke R., LeBlanc M., Zubek R. (2004) *MDA: A Formal Approach to Game Design and Game Research*. AAAI Workshop - Technical Report. 1. Available from: https://users.cs.northwestern.edu/~hunicke/MDA.pdf [Accessed 19 May 2021].

Iida, H. (2007) *On games and fairness*. Available from: https://www.researchgate.net/publication/309392093_On_games_and_fairness [Accessed 18 May 2021].

Ironhide Game Studio (2011) *Kingdom Rush* [game]. Available from: https://store.steampowered.com/app/246420/Kingdom_Rush___Tower_Defense/ [Accessed 18 May 2021].

Ironhide Game Studio (2013) *Kingdom Rush Frontiers* [game]. Available from: https://store.steampowered.com/app/458710/Kingdom_Rush_Frontiers__Tower_Defense/ [Accessed 17 May 2021].

Ironhide Game Studio (2018) *Kingdom Rush Vengeance* [game]. Available from: https://store.steampowered.com/app/1367550/Kingdom_Rush_Vengeance__Tower_Defense/ [Accessed 18 May 2021].

Macklin, C. and Sharp, J. (2016) *Games, Design and Play: A Detailed Approach to Iterative Game Design*. 1st edition. Boston: Addison-Wesley Professional. [Accessed 16 May 2021].

Robot Entertainment (2012) *Orcs Must Die! 2* [game]. Available from: https://store.steampowered.com/app/201790/Orcs_Must_Die_2/ [Accessed 18 May 2021].

Royce W.W. (1970) *Managing the development of large software systems*. Available from: http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf [Accessed 19 May 2021].

Schell, J. (2015) *The art of game design: a book of lenses*. 2nd edition. Boca Raton, Florida: CRC Press. [Accessed 16 May 2021].

Taylor, D. (2013). *Ten Principles for Good Level Design* [video]. Available from: https://www.youtube.com/watch?v=iNEe3KhMvXM&ab_channel=GDC [Accessed 15 May 2021].

Trendy Entertainment (2017) *Dungeon Defenders II* [game]. Available from: https://store.steampowered.com/app/236110/Dungeon_Defenders_II/ [Accessed 18 May 2021].

Tynan, S. (2013) *Designing Games*. Sebastopol, CA: O'Reilly Media, Inc. [Accessed 18 May 2021].

VOiD1 Gaming (2020) *Flow Theory in Game Design.* Available from: https://www.void1gaming.com/post/flow-theory-in-game-design [Accessed 13 May 2021].