

Design by Kyle Harrison

For Xbox Live Arcade and Windows

**ESRB Rating**: Teen

### Index

**Content Page**

1. Game Design Overview 3
2. Story 4
3. Game play 5
4. Elevator Pitch 5
5. Player Character 6
6. Player Controls 6
7. Game World 7
8. Game Experience 7
9. Phase 1 – Build 7
10. Phase 2 – Resources 8
11. Phase 3 – Boss 8
12. Game Mechanics 9
13. Resource Mechanics: 10
14. Spells 11
15. In game events 12
16. Enemy units 13
17. Friendly unit’s 14
18. Cut scenes 15

Software and Hardware

1. Software and Hardware Justification 16
2. Software 17
3. Hardware 18
4. Software and Hardware Configuration 18

Graphics and Audio

1. Graphics and Audio 19
2. Graphics 19
3. Graphical Style 20
4. Graphic source 20
5. Audio 21
6. In-Game Audio 21
7. Source of Audio 21
8. Cinematic Cut Scene 22

Game Specific Features

1. Real time Game play 23
2. Collision Detection 24
3. Artificial Intelligence 24

Game Code Structure

1. MVC Design Pattern 25
2. Model Layer 25
3. Controller Layer 26
4. View Layer 26
5. Strategy Design Pattern 27
6. Singleton Design Pattern 27
7. State Design Pattern 27
8. Observer Design Pattern 28
9. UML diagram 29
10. Behaviour diagram 29
11. Time management plan 30
12. References 31
13. Appendix 32+

Ariyana

Genre: Strategy Horror

Target Audience: Age: **12+** / ESRB Rating: **Teen**

Platform: Windows/ Xbox Live

**The Big Idea/Concept**

Ariyana is a noir style world inspired by the two opposing sides in chess whilst incorporating elements of tower defence and real time strategy elements to provide a unique twist on a traditional game. The boundaries between black and white blur drastically when good must verse evil in the ultimate showdown where only one side can claim victory of the war and rule the land!

Game Summary

As the last line of defence, the inhabitants of a peaceful realm take a stand when Oroan, a demon lord threatens all life in the kingdom! Gathering armour, weapons, and even prayers holy soldiers are summoned to defend the last human stronghold against an unrelenting destructive wave of demons.

Ariyana is the master tactician controlling unit formations on the battlefield. Collecting and storing Mana will allow holy units to be summoned to the battlefield and battle morale will influence how units behave upon the battlefield.

### Play Mechanic

Players create and manage a grid type battlefield with a variety of unit types and resources in real time. Enemies attack periodically in waves and waves increase in size over time throughout three phases. The player gathers resources during the game to summon additional units to the field. A spell book grants access to special abilities and allows the mechanics of the game to fit in with the story.

### USP (Unique Selling Points)

* Easy to play, Hard to master ( Increases replay Value )
* Summon and manage units for battle formations (Lots of unique classes with possibly of more in the future)
* Wizards chess and Tower defence with a twist ( familiar concepts, new way to play)
* Fight increasingly difficult waves of demons and defeating the boss to unlock the end of the story.
* Unique graphical style not seen in very many other games. (Emphasising the divide between good and evil)

### Similar competitive products:

Plants vs. Zombies, Wizards Chess, Tower Defence

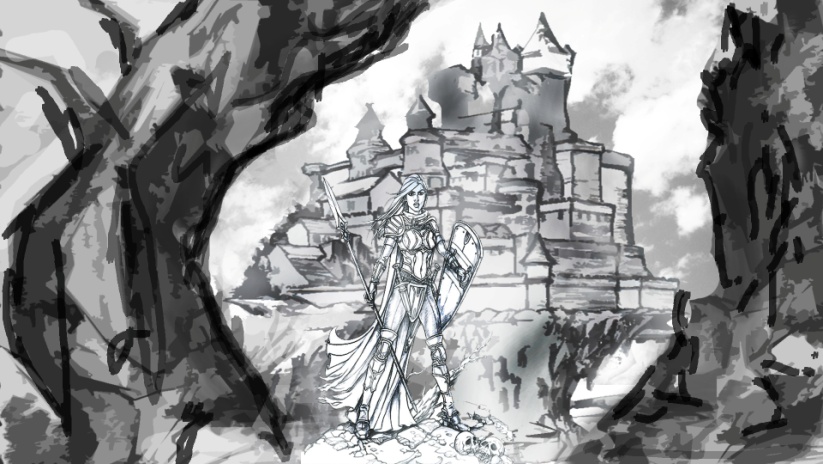
### Game Story:

The kingdom of Ariyana was not always a peaceful realm as millennia ago the land was scourged by Demons that sensed the life force from the netherworld. During this time everything in the realm descended into a great darkness for a full lunar cycle causing the gates bordering this realm from the netherworld to weaken and the demon’s to break through. Divine authority’s seen this that later became to be known as gods put a stop to the rampage of the demon’s and returned the land to its original state however they were unable to stop the boundaries from remaining weak where the demons once crossed so they build a fortified castle and selected a chosen one imbued with their greatest strengths as a reminder and defence against the dark ones should they ever return.

Now in the present day, after years of imprisonment for using the Dark Art’s to gain power and breaking ancient laws set in place by the Gods of the land; Oroan, the Demon Lord has escaped from his prison and opened a portal to the netherworld to unleash his rage and havoc using the encroaching darkness lingering on the other side to exact his vengeance on all those that attempted to stop him.

As the last line of defence the inhabitants of the realm take a stand when Oroan opens the portal to the netherworld tearing the land asunder and threatening all life in the kingdom! Gathering armour, weapons, and ancient prayers these once peaceful inhabitants will be forever changed into holy warrior’s seeking only retribution for their corrupted land and the fallen consumed by the demonic darkness unleashed. Now they make their final stand on the frontlines of the battlefield to defend the last human stronghold against an unrelenting destructive wave of demons!

You are Ariyana the master tactician! Forced into a life of servitude against your will, raised in a heavily fortified castle with nothing but the company of elders throughout the year, and knowing nothing but war and anger your entire life; you have doubted your purpose as the chosen one for many year as have many other’s but your time has come to rise up and perform your duty! Instilled with great power and divine knowledge through years of intense religious training you have mastered the art of accumulating Mana from the heavens allowing you to summon a divine infantry and commanding those that look to you for guidance from the precipice overlooking the battlefield.

Using spells from an ancient tome passed down through generations you alone control the outcome of the battle and the fate of all those under your watch. Using your divine powers you guide the holy warriors on the battlefield by influencing their morale and fate alongside maintaining your own faith in a particular god bestowing unique blessings upon the infantry under your command. What most people would consider a blessing, you see as a curse until the moment you are needed most by your people. You are the chosen one and now is your moment to rise!

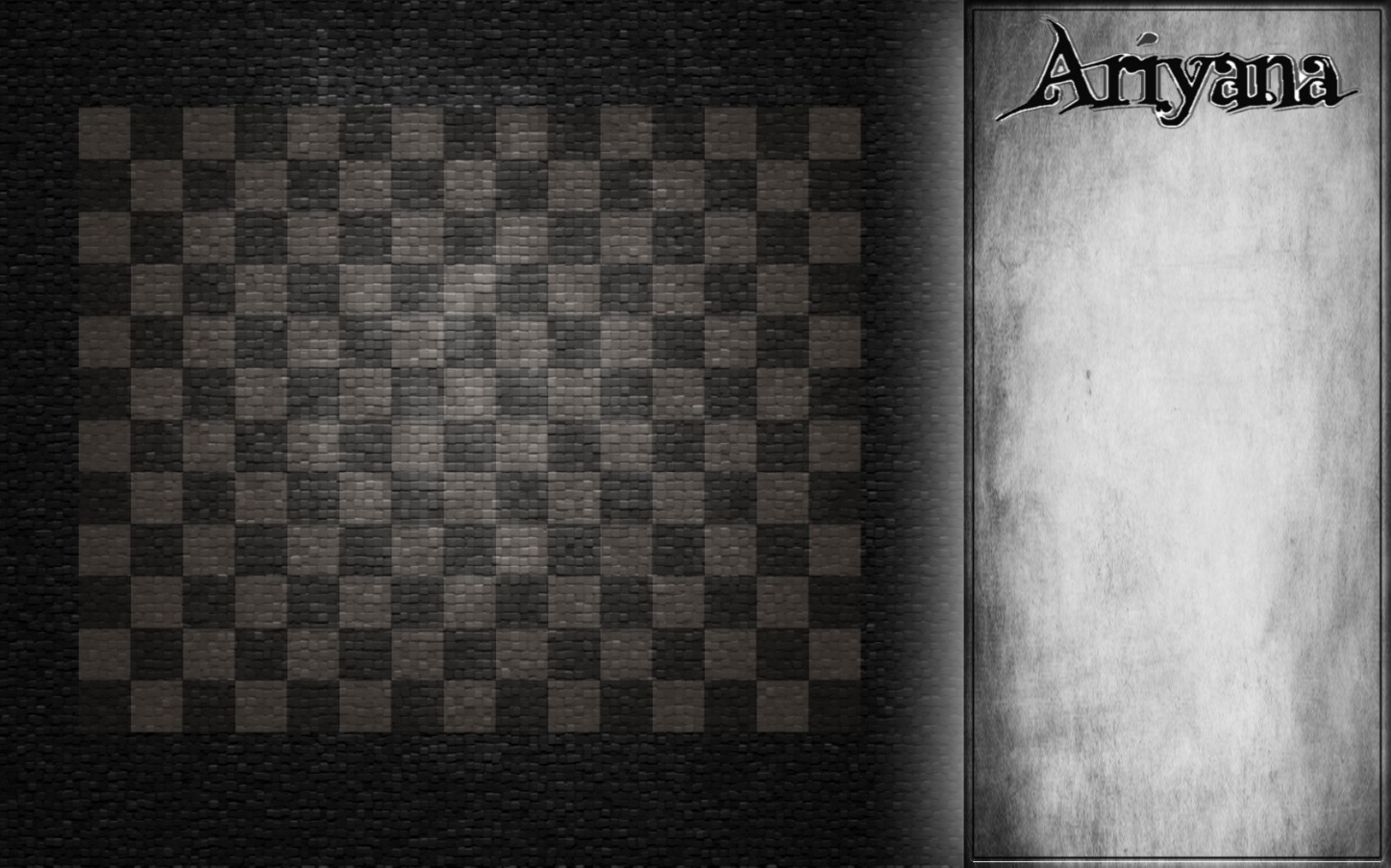
### Game play:

In Ariyana, the player is a female paladin who is defending her castle from demonic entities. The game play is separate into three distinct phases each with a limited time of play. The player deploys a range of friendly combat units onto the battlefield to defend the castle from being breeched by demons and protecting the innocent inhabitants inside. The player defends the castle from several waves of enemies and a boss using their customised battlefield.

Each game play round lasts approximately 10 minutes and the three phases will have to be completed in this time to win the game. Finishing the game with time remaining will generate a higher score. The three phases will consist of a build phase; defend phase, and a boss phase. The build phase will offer smaller waves of enemies and will last 2 minutes. The second phase is the defend phase and this phase will last 5 minutes with increasing waves of enemies that scale in size and difficulty. The third phase is the boss phase and will offer a small build phase followed by a large increase in enemy’s waves followed by a final boss that must be defeated to win the game.

During all phases the player deploys and commands unit placements on a grid type battlefield using an arsenal of unit types that have varying effects on how the game is played. Enemies attack periodically in waves and these encroaching waves increase in size over time and have different rules for spawning during each phase. The player gathers resources automatically during the game to summon units to the field and whenever they defeat an enemy.

During key points in the game the player will unlock new units in the spell book when he/she reach a certain level of experience which is gained from killing enemies. Beneficial spells will be available in the spell book which will be subject to cool down timers to balance the game. A resource that will be gained or lost during the game will be battle morale and maintaining this will influence how units act upon the battlefield in terms of how fast they can attack and how much damage they can produce.



### Elevator Pitch:

Chess meets tower defence set in a medieval fantasy world.

***“On the Battlefield when black clashes with white there is no room for grey!”***

### Player Character:

**Ariyana**: Ariyana is the antagonist in the game however you never see the character and her portrayal in the art work is for story telling purposes. A starting Ariyana player will have access to basic unit types and a single spell. As game play progresses more units and spells will become unlocked as experience is earned from killing demons in that round which can be used to balance the game when the difficulty increases. The player elects how he/she wants to spend their resources and influence how the game is played by the choices they make.

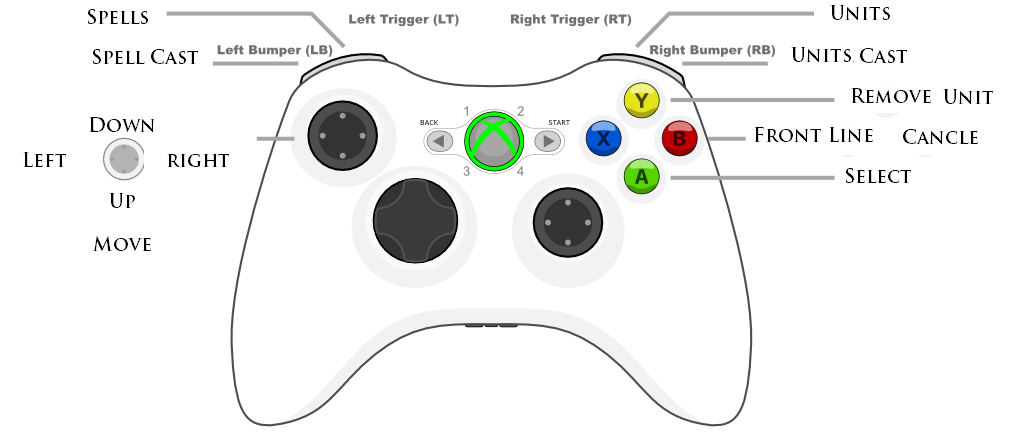


Resources such as spells have a cool down and the player may only be able to cast this once so strategy plays an important role as overwhelming yourself at the start of the game may require choices to be made that will later influence the strategy you need to complete the game. Each unit type unlocked will have its own advantages and disadvantages. It is the job of the player to decide the best formations to use and how to structure the battlefield for maximum efficiency and how to beat the game. Once the player has learned how to beat the game it is then their goal to produce the highest score possible in a single round.

The player’s view of the board will be a top down view as if the player is on a precipice viewing the battlefield and issuing commands. The player will be able to view spells and units available at any time during the game however it will only be able to cast if the resources are available. All resources will be displayed on screen and can be used at anytime.

### Player Controls

The player controls will utilise an Xbox 360 controller. The controller will not make sure of all the buttons as they are not needed however instead of having these buttons not perform anything they will be mapped to functions already present such as the D-pad will provide movement through the game board and menu systems.

The controller will use a traditional mapping system of movement on the left with the left analogue stick/ D-Pad and frequently used controls on the right using the A, B, X, Y buttons that are easily accessible with the right thumb. The triggers will access the Spell/Unit menu and the bumper buttons will cast the spell. As the menu will be accessed more than a spell or unit will be cast these have been assigned to the triggers and the cast for each has been assigned to the bumper as it is less likely the player will press the bumper when they do not want to perform a cast.

### Game World

The world of Ariyana is inspired by the fictional game of Wizard’s Chess from Harry Potter which brings the classic game to life with fantasy chess pieces that are animated and literally destroy or capture each other if they land on an opponent's square. Alongside this it takes elements from Tower defence games such as Plants Vs Zombies in which an open grid type game board is used instead of a pre determined path. The graphical style inspirited by classic film noir movies such as “Nosferatu” and uses black and white graphics to emphasis the divide between good and evil.

The object of the game is to defeat multiple waves of enemies and an end level boss by placing units on a chess like game board. A player attempts to defend a castle until all enemies are defeated or an enemy breaches the gates ending the game. The battlefield is created to be extremely customizable using a variety of unit types to defend the castle but of a size that is also manageable to allow players to quickly determine which pathways on the grid are currently under attack and redeploy their units appropriately. Players can change the formation of units on the battlefield at anytime to ensure the best defence and offensive strategy is in play.

Incremental waves of enemies spawn and move towards the castle to attack it and all that stands in their path. These waves are vary in the three phases and various wave types comprised of different enemies attack within a set amount of time allows players to power thought waves of enemies using an offensive or defensive strategy allowing multiple play styles and multiple strategy’s for winning the game.

### Game Experience:

### Ariyana is divided into a 3 phase event in which the player must complete all three phases in succession to win the game. These phases will automatically progress when a timer increments past the allocated time for that phase regardless if the player is ready or not to enter the phase. This will increase the difficultly of the game as good and bad decisions will later effect the outcome of events. The game will have a learning curve that the player will experience though multiple play thoughts and attempts. The player may easily pass stage 1 to find that the choices they have made have hindered the progress of stage 2. The three stages that will create a round of game play is the build phase, resource management phase, and boss phase.

### Phase 1 – Build Phase

Using the spell book, the player can perform spells casts to summon units to the field or activate beneficial spells. A range of unit’s will be available during the build phase with more becoming unlocked in the resource management phase. The goal of this phase is to build up an initial fighting force and prepare for the remainder of the game. Unit’s can be moved at anytime and removed from the battlefield returning a small portion of their original cost to the player’s resources. This phase will start with small waves of enemies that gradually build in size and toughness lasting a total of 2 minutes before progressing to the resource management phase.

**Events the occurring during phase:**

* **Available Units** - A number of units are to be selected and placed on the battlefield from the Fighter/Spear-man unit types.
* **Moving Units** – A number of unit’s will be required to be repositioned on the battlefield to successfully defend about the enemy AI.
* **Defend Castle**- The player is required to have at least 1 unit alive at the end of the phase to successfully defend the castle.
* **Battle Morale** – If at any time all unit’s on the field are dead the enemy AI instantly wins when morale drop’s to zero.
* **Stage Progression** - Once the player has successfully defended the castle for 2 minutes the resource management phase will commence.

### Phase 2 – Resource Phase

A tactician must tend to the infantry under their control and manage their resources well and the player is no different. In chess there are disposable pieces and pieces that are essential in winning the game. It’s only when you learn how to use these to their best effect do you truly understand the nature of the game. At the start of this phase 2 new units will be unlocked that provide a defensive option to the player to ensure they do not get crushed by a wave of demons.

These defensive units will aid in creating a barrier between the warriors on the battlefield and demon’s reaching the castle by slowing down demon progress. In this phase resources will be in abundance for the first 2 minutes and then an evil curse will be cast that will slow the generation of resources. In addition waves will increase in size and difficulty allowing enemies to deal more damage and also have a greater level of toughness against standard unit attacks. When the curse is cast the player will have to utilise their infantry on the battlefield to maximum effect and make use of all classes to maintain the upper hand. In this phase battle morale will come into fruition as letting too many units die even though a defensive is still being maintained will cause units to suffer from a debuff effect in which their attack and defence are reduced making them weaker against enemies.

**Events the occurring during phase:**

* **Units unlocked:** The Shield bearer, Archer, and Healer unit class is unlocked in the spell book.
* **Battle Morale:** – Whenever a unit dies the battle morale on the field is lowered and attack and defence values of units are dropped.
* **Evil Curse** – After 2 minutes in this phase an evil curse will be placed upon the land causing a debuff in which all resources generated will be limited by a percentage for the remaining time of the phase.
* **Stage Progression** – The phase will last 5 minutes with waves of enemies scaling in size and difficulty.

### Phase 3 – Boss Phase

In the third phase the player will face even larger waves of enemies with the conclusion of the fight being the boss in which the player will have 1 minute to defeat the boss before he makes his way to the castle. At the start of the phase where will be a boost in resources and the player will be able to quickly rebuild their infantry on the battlefield before they are over run. This is a temporary buff and wears off quickly or when all of the resources it allocates is consumed. This buff allows the player to summon the new mage class which is unlocked during this phase to the battlefield and replenishing the defences. Meanwhile the enemy demons will try to wear down the defences as much as possible so Oroan the demon lord boss is able to quickly push through those that remain to defend the keep.

**Events the occurring during phase:**

* **Final Unit Unlocked** – Mage class is unlocked.
* **Here comes the cavalry** – Resources are instantly increased and made bountiful for a short period of time
* **Demon Lord Summoning** – The demon lord takes 1 minute to be summoned. During the summoning a large wave of demons attack in a final push to weaken the defences as much as possible before the boss arrives.
* **Demon Lord Arrives** – The demon lord is summoned to the battlefield. The player has 1 minute to defeat the demon lord before the game ends. If the demon lord is undefeated in this time or if all units die and a demon or the demon lord beaches the castle the game is over with victory going to the demons.

### How to win:

To win the game the player must survive and defeat all waves of enemies including the boss within the 10 minute time frame. The first 9 minutes require the player to defend the castle, manage resources, all whilst managing units on the battlefield and throughout the three phases until the last minute of game play when the Oroan, the demon lord (boss) enters the game who the player must defeat before he beaches the castle or time runs out.

### Game Mechanics:

Game play in Ariyana is in the form of a top down view of 2D characters and the battlefield alongside a view of the HUD and the spell book seen in image of the layout below. The camera is in a fixed position giving the player a view over the entire battlefield alongside a view of the spell book on the right hand side of the screen. The player views the battlefield in real time viewing enemies progressing downwards towards the castle and units under the player’s control defending against the oncoming demons. The player has the ability to move the units under its control to a fixed number of squares on the game board at any time as long as the square is not already occupied by another unit or enemy.



The game mechanics can be broken down into 5 sub types of mechanics that will merge together to form the game.

**Resource mechanics**

The HUD displays the experience gained, the score of the player, the current phase and time remaining, and the battle morale of the unit’s on the field. These resources will continuingly change throughout the game

**In game events**

These special in game events will breaks the normal game flow by introducing new elements that will make the game easier or harder depending upon which event is activated.

**Units**

These will be the warriors the player places on the field to defend the castle. There will be a selection of units available to choose from however more will become unlocked as the game progresses through phases.

**Enemies**

The enemies will be comprised on demons with various characteristics that will attack units controlled by the player whilst attempting to besiege the castle.

**Spells**

These are special in game action’s that have beneficial effects for the player however they can only be used when they are not on cool down and they are only available once unlocked by the player though gaining XP.

### Resource Mechanics:

**Time and phase system:** The HUD will display the current time remaining that will decrement each second when the game starts. In addition to this the current phase will also be displayed to the user with a progress bar moving along the phase bar to show how far the player is from completing the stage.

**Mana :**Mana is generated at a rate per second and boosted by 1% for each enemy killed. As the Mana bar fills the image will change from dark to light and vice versa when it is drained.

**Score system:** Generation of points will be calculated automatically based on a player’s actions in the game. Points will be accumulated for the death of enemies, whenever a phase is successfully completed, and when a player completes a time challenge such as vanquishing 6 waves per phase. The points will be displayed in a bar on the HUD which will update through a draw cycle every second.

**Gaining Experience:** In-gameSpell’s unlock whenever the player gains a set amount of experience. These spells have cool downs and may only be used once or twice a game however they will be beneficial to the player and have the ability to turn the game around. The experience gained from each demon death will fill an experience bar and at key points in the bar these spells will unlock in the spell book. The amount of experience will vary per demon and more difficult demons to kill will grant more experience. However experience will be deducted if unit’s die at a fixed penalty of 1 experience point per unit death. This penalty urges players to keep units alive as long as possible

**Battle Morale:** Battle morale will initially start at neutral in phase one and no changes will be made to the statistics of a unit’s power/toughness until the battle morale reached a fixed percentage increase in to determine either positive or negative battle morale. The battle morale bar will be visible on the HUD and will initially be displayed as a neutral gray bar. As the player progresses through the game and their actions produce changes then their battle morale changes to reflect this.

During game play this will change to a filled bar and vice versa, so if morale drops it will progress to the darkened bar displayed below. This morale serves as a reminder in the game that. The problems progress will be determined in light or darkness. This shows which side is winning as the screen brightness will alter between light and dark for the two opposing sides.

**< Low Morale | Neutral Morale | High Morale >**

****

****



### Spells

The following is a list of spells that can be cast whenever they are unlocked during game play. Spells are subject to a Mana cost which is required in order to cast the spell and a cool down effect which only enables the spell to be cast once before requiring a cool down period before it can be used again. Spells are unlocked at different levels of experience gained which stops a player from becoming overly powerful at the start of the game and encourages progression through the game content.

**Call to Arms**

****

**Description**: Ariyana unleashes a battle cry inspiring all unit’s on the battle field and increasing battle morale. Units that hear her battle cry fight harder and survive longer for a short time allowing the tide of battle to turn in moments when morale falls low.

**Effect**: Increase morale for a short time. Effect is greater when morale is low.

**Holy Prayer**

**Description**: Ariyana uses her divine powers to summon healing energy from the heavens to restore the life force of a unit on the battlefield weakened by battle. The unit healed will maintain the residual energy from the holy prayer remaining empowered by the energy for a short time.

**Effect**: Instantly increases a unit’s health and continues to increase afterwards for a short time.

**Divine Protection**



**Description**: Ariyana calls upon the heavens to open protecting all those in its light for a short time whilst causing the demons to burn with holy fire whenever a unit is attacked on the battlefield.

**Effect**: Reduces all damage taken by units by 80% and negating 20% of demon base damage to demon as holy fire damage. Has no effect against Oroan, the Demon Lord.

**Cleansing Fire**



**Description**: Ariyana unleashes all her might and calls forth a cleansing fire to engulf all enemies on the battlefield effectively purging the land of all demons.

**Effect**: Purges the land with a cleansing fire damaging all demons by 90% of their base health that are currently active on the battlefield. Has no effect against Oroan, the Demon Lord.

### Spell Summary Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Spell** | **Mana Cost** |  | **Cool down** | **Ability** |
| **Call to Arms** | 100 |  | Once every 2 minutes | Increases morale for a short time |
| **Holy Prayer** | 50 |  | Once every 60 seconds | Heals a unit to full health |
| **Divine Protection** | 150 |  | Once every 5 minutes | Summons’s a temporary barricade for 30 seconds |
| **Cleansing Fire** | 250 |  | Once per game | Damages all demons on the battlefield by 90% base health. |

### In Game Events

The following is a list of in game events that will be triggered automatically during phases of the game. These events will implement event specific rules in addition to the rules of the current phase. These events serve the purpose of changing how the game is played by making the game difficult in times when the player has had a chance to gather resources and structure the battlefield with units alongside this they also help the player at times when game play suddenly changes such as massive waves of difficult to defeat enemies require the player to have additional resources to spend. They also allow the player to establish how far they are into the game by reaching set points of game play such as a phase progressing to the next phase.

**Phase Progress**

**Description:** The warriors under your command have fought with great honour showing the inhabitants of the castle that they fear no demon and protecting the castle from attack. They stand their group and prepare for the next phase of demons to approach.

**Effect:** Progress to the next phase of the game

**Evil Curse**

**Description: A**n evil curse befalls the land causing the skies to darken and as a result all resources begin to dwindle and become harder to produce as the heavens are out of sight. As morale drops it is faith and strategy alone that is the key to survival.

**Effect:** After 2 minutes in phase 2 all resources generated will be reduced by 80% until the end of the phase.

**Here Comes the Cavalry**

 **Description**: The warrior’s on the battlefield have fought valiantly after suffering great losses. Inhabitants of the realm witness these courageous acts’s and become inspired to devote all their remaining efforts to producing resources desperately needed and lifting battle morale.

**Effect:** During Phase 3 a player’s Mana pool and battlefield morale are instantly increased boosted for a short period of time.

**Demon Lord Summoning**

**Description:** The demon’s sense their lord is approaching. Regaining their strength they make a final push for the castle to destroy it before their master arrives. There is no holding back for the demon’s anymore.

**Effect:** During Phase 3 It takes 1 minute for the demon lord to be summoned. A large wave of demons attack in a final push to weaken the defences as much as possible before the boss arrives.

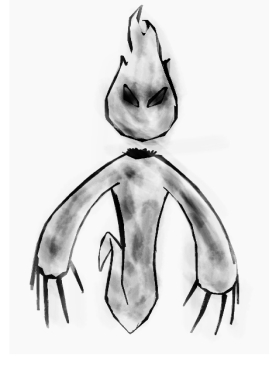
**Demon Lord Arrives**

**Description**: The demon lord is summoned to the battlefield giving the player 1 minute to defeat him before the game ends. If the demon lord is undefeated or if all units die and a demon or the demon lord beaches the castle the game is over with victory going to the demons.

**Effect:** During phase 3 the player has 1 minute to defeat the demon lord or the game is lost.

### Enemy Units

**Flame Elemental**



**Description**: The flame elemental demon is spawned at the start of the game. It is a weak demon that is easily defeated due to its low attack power, low range, and low defence. The flame elemental can only attack units in the space directly in front of its path.

**Special Ability:** It has a special ability that is activated when it dies called Immolate. The effect of this ability is triggered upon its death and it deal’s a small amount of damage to all surrounding enemy units.

**Wraith**

**Description:** The Wraith demon is spawned during the first phase when the player has had a chance to formulate a defence. The Wraith will appear in small numbers at first but quickly large swarms will form that left unopposed will quickly destroy all in their path with stronger attacks, greater defence, and longer range.

**Special Ability:** The Wraith has no special ability as its toughness and ability attack from a range of 2 squares enables it to attack units from further away. Against Fighters the wraith is vastly superior and can quickly and easily dispatch a unit from a distance.

**Demon Skull**

**Description:** The Demon Skull is spawned in the second phase as the purpose of this demon is to lower the defences of unit’s on the battlefield to make them weaker prey for other demons. It has a medium attack but a weak defence so its best form of attack is from a distance of 3 spaces and it is strong against Fighter, spearmen, and shield bearer units.

**Special Ability:** The demon skull can lower the defences of unit’s making them vulnerable to its medium powered attacks before units can eliminate it. The skull attacks alone with larger waves of enemies that provide protect whilst aiding those that protect it.

**Hell Spawn**

**Description:** The hell spawn is spawned in the third phase and the purpose of this demon is to destroy and devastate the battlefield as much as possible before Oroan, the demon lord arrives. This demon has a long attack range and can easily use its high powered attacks to kill units from this great distance.

**Special Ability:** The hell spawn demon can devour those that fall victim to it. With each fallen warrior the demon grows stronger and must be killed quickly before it becomes too powerful. These demons can quickly change the outcome of the game if the player does not step in to stop their special ability from wiping out all units on the battlefield.

**Demon Lord**

**Description:** This is Oroan, the demon lord who is summoned at the end of the game. Once human it can be seen that any trace of humanity is long gone and this is reflected in the destructive power of Oroan. With a high attack and defence this demon takes considerable force to kill and has the power to instantly end the game if not killed within 1 minute of spawning.  
  
**Special Ability:** Oroan’s special ability is to consume all living units on the battlefield within 1 minute of spawning. This is effectively the end game challenge as failing to kill Oroan ends the game and causes the demons to win.

### Friendly Units

**Armoured Fighter**

****

**Description:** The armoured fighter is the first unit available in the game. It is the most basic unit however it offers a high attack and medium defence against enemies. Having an attack range of 1 square it can only attack enemies directly in front of it however it is vulnerable to enemies with ranged attacks. Used well in combat armoured fighters can be a formidable foe to any enemy.

**Special Ability:** No special ability due to high attack power and defence

**Spear man**

****

**Description**: The spear man is a basic unit available in phase one. Unlike the armoured fighter he has a weaker attack but stronger defence and has the ability to attack from a range of 2 spaces.

**Special Ability:** The spear man has no special ability as he is a basic unit in the game and is already balanced to provide a medium attack and defence against enemies.

**Archer**

****

**Description:** The Archer is the first long distance defender available in the game. The archer has a range of 4 spaces making it able to attack enemies from a greater distance. The archer can attack from behind the front line making it able to defend other units without putting itself in danger.

**Special Ability:** The archer’s special ability is the multi shot which attacks enemies up to 4 spaces away in 3 lanes allowing a medium attack to be delivered to three enemies at once.

**Shield Bearer**

****

**Description:** The shield bearer is a defensive unit that is available in phase 2. The shield bearer has no attack instead opting to focus on having a high defence to stop enemies from breaching the front lines. This unit relies on those behind it to dispatch the enemies with ranged attacks.

**Special Ability:** The special ability of the shield bearer is a shield slam which pushes the enemy back 1 space making it travel forward again before attacking.

**Healer**

****

**Description:** The healer is a defensive unit which becomes available in phase 2. The healer has no attack or defence and instead ops to help those around her by healing their wounds.

**Special Ability:** The healer’s special ability is to heal all units around her within 3 spaces by 1% of their base health per second.

**Mage**

****

**Description:** The mages is a purely offensive unit with a strong attack and low defence that is available only in phase 3. A mage used effectively can provide a great offense.

**Special Ability:** The mages special ability is to cause Immolation which causes all enemies hit by the mages attack to ignite and become engulfed in flames delivering 2% damage per second.

### Enemy Summary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unit** | **Range** | **Damage** | **Defence** | **Ability** | **Phase** |
| **Flame Elemental** | 1 Space | Low | Low | Immolate – Deal’s a small amount of damage to all enemies around Flame elemental dies. | 1 - Build |
| **Wraith** | 2 Spaces | Medium | Medium | None | 1 - Build |
| **Demon Skull** | 3 Spaces | Medium | Low | Yells Obscenities – Lowers enemies defences | 2 – Resource |
| **Greater Demon** | 4 Spaces | High | Medium | Devour – Whenever a greater demon kills a unit it grows stronger. | 2 – Resource |
| **Oroan, The Demon Lord** | 3 Spaces around him | High | High | Consume all – Within 1 minute of spawning Oroan kills all units on the field. | 3 – Boss Fight |

### Friendly Units Summary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Unit** | **Range** | **Damage** | **Defence** | **Ability** | **Cost** |
| **Fighter** | 1 Space | High | Medium | None | 75 |
| **Spearman** | 2 Spaces | Medium | Medium | None | 125 |
| **Archer** | 4 spaces | Medium | Low | Multi shot | 150 |
| **Shield Bearer** | 1 Space | None | High | Shield Slam – pushes back 1 space | 300 |
| **Healer** | 3 Spaces around | None | None | Heal – heals all in area by 1% per second | 450 |
| **Mage** | 4 Spaces | High | Low | Immolate – Causes all enemies to ignite and become engulfed in flames 2% damage per second. | 500 |

### Cinematic Cut Scene

The game will contain two cut scenes that will be used to deliver the story to the player. The opening cut scene will be presented to the user before the game starts to establish where the player currently is in the story line. The closing cut scene will be used to finalise game play and deliver the ending story to the player. Both cut scenes will allow the player to skip through if they do not want to view the story and want to get straight into game play. If the game is returned to later more levels could be added and these cut scenes could serve as the story in-between new levels and grid types however for now a single self constrained story will be used.   
  
The opening cut scene will provide the player with an introduction to the story without the need to read a large paragraph of text before starting the game and will focus on delivering the entire game story listed in this document in a short action packed cinematic. This cut scene will show a land ravaged by demons, the battle taking place between units and demons on the battlefield, and Ariyana standing in a defensive stance in front of the castle. All images will be created in black and white using as much concept art as possible with effects to add animation.

The closing scene will provide a firm conclusion to the game allowing the player to end the story. The aim of the game is to make it difficult for the player to simply pick up and power their way through the phases and instead requires the player to think about how to effectively win the game. The closing cut scene will show the land being saved from demons and show the causalities on the battlefield along with those that survive followed by a cliff hanger at the end to leave the cinematic story open to extension.

### Software and Hardware

Everyone loves casual games. They are easy to pick up and play for 5 minutes then walk away however developing a title and making it available in a market place saturated with casual games is no easy task and this has been taken into consideration when selecting the hardware and software to be used for developing and eventually releasing “Ariyana” on a specific platform. The route to market is often an important question to ask and for this game concept to stand a chance of surviving in a overly saturated market place with other casual games the two most viable platforms for developing this type of game is an Android and the PC/Xbox platform.

However after much consideration of both platforms the decision to use Microsoft hardware and software for developing Ariyana was made because of the following reasons:

**Release on Xbox Live Indie Games**

Initially the appeal of creating a game that would be available for download on for Xbox Live Indie Games was tempting as it was an established market with a low barrier for entry however it benefits from having some restrictions on entry unlike the android market place where anyone could publish an App regardless of quality and provides a viable route to market that is not as heavily saturated as the Android market place.

**Access to freely available Software**

The software needed to make games for XBLIG is freely available. The benefits of XBLIG over releasing on Xbox Live Arcade are that XBLA requires an official developer application which costs thousands of pounds and the game could still be rejected. For an initial game release XBLIG is ideal as it is tailored to hobbyists and first time developers getting their game released on a major videogame console**. (1)**

**Developing for specific standard hardware**

PC and Xbox hardware is easily accessible and as the Xbox platform does not change in terms of hardware it is easier to develop games for this platform when viewing development in terms of hardware and software. As for PC hardware even through there is a plethora of hardware and software available even the most basic computers are capable of playing some form of video game and as this is a casual game it will not be overly demanding upon the system and should be widely accessible.

**Learning Curve of XNA**

Having previous experience using the C# programming language and the Visual Studio environment this decreased the learning curve of developing a game allowing for greater time to be spent on developing a game of a higher standard and quality instead of learning a new programming language and environment in a restricted time frame.

**Cross Platform Gaming**

The benefit of developing a game in XNA is it’s easy to create a cross platform game that is playable on both Windows and Xbox using a gamepad or other standard hardware with minimal code changes.

### Software

The following is the software that will be used when developing Ariyana. Some of this software will be used to create the game and other software will be used to create graphic and audio elements of the game.

Visual Studio 2010/XNA 4.0

Developing for both the Pc and Xbox to making a cross platform game within Visual Studio and XNA was the only option as official developer application tools cost thousands of pounds and would not be viable for a project of this undertaking.

XNA has been around for a while and in that time it has went through various iterations and become the standard for indie game developers on devices such as the PC, Xbox, Windows Phones, and Zune as it allows anyone to release a small game and have millions of people around the world play it, and also allows for small amounts of revenue to be made through selling the game. In addition it is suitable for beginners learning games development as Xbox Live Indie Games is the ideal place for hobbyists and first time developers to release their title to a marketplace and gaining feedback from an active community.

The main benefit of XNA to a first time developer during development is that there is no need to worry about low-level hardware engines as XNA provides a framework and elements of a game engine through the XNA library. This library allows a developer to quickly produce a game and create a prototype which can be developed into a demo or even the fully finished game instead of building an entire engine to before seeing any results. Taking this step up in the level of abstraction allows more time to be spent on development resulting in a better game being produced as less time is spent worrying about how to make something work and time is spent on polishing the game. In addition XNA allows for rapid development of an entire game or even larger tech demos allowing for a game to be delivery in a short time frame.

C# Programming Language

Minimising the learning curve of making a video game for the first time by using a familiar language aids in the development process as additional time is not needed in order to learn a new language. This time can instead be used to further development in other aspects of the game. This transferrable skill allows for experience and exposure to the language gained previously to be applied and learning to commence and continue at an earlier stage in the process which is essential in an undertaking such as this where time is a limited resource.

The build process within XNA allows for developers to get the basics up and running without having to worry too much about large portions of development time being devoted to this however on medium to large scale projects this build process aids in having the ground work laid out and will aid in creating a more complex build process. **(2)**

### Photoshop

To produce graphics for the game quickly and efficiently any in game art work that I produce myself will be created or edited from line drawings in Photoshop. This allows art to be quickly to be sketched on paper or taken from other artists and easily modified to the art style of the game. In addition to producing art it will also be used to create portions of the interface alongside the sprites that will be used in game.

### Audacity

Audacity will be used to record sounds and edit pre recorded and existing audio. Audacity has a small learning curve and offers a range of options that can be used to modify audio to better suit the needs of the game. Some voice over effects that will be used in the game alongside sound effects will be recorded in Audacity and then edited to produce clear and audible recordings that fit the style of the game.

### Hardware

Ariyana does not require a great deal of processing power as it is being developed as a casual indie game offering 2D black and white graphics however the standard hardware configuration that comes with a Xbox 360 is more than capable of running the game and most modern computers even with basic specifications are capable of playing almost all casual games.

XNA offers a range of input devices other than the standard keyboard and mouse for PC such as Official Xbox controllers and other gamepads alongside Xbox Kinect of which are all compatible with both PC and Xbox 360. However for this project I will be using the Xbox gamepad to control both PC and Xbox versions of the game. Gamepads work exceptionally well with XNA and using a gamepad in both versions allow for the game to port easily between both platforms however in the PC versions keyboard and mouse controls may be included also as users may not always have access to a gamepad.

The justification for using a gamepad for this game is that an Xbox controller is a standard device in which people are familiar with and have come to expect a certain control scheme based on hand placement. It is easy to learn the controls of a game without reading a manual as mapping actions to specific buttons or trigger allows a user to learn quickly through trial and error. As the game will rely on simplistic controls not all buttons or triggers will be needed however instead of having these buttons do nothing they will be mapped to similar actions as other buttons so that a player who accidently pressed a button will trigger an event instead of nothing happening and the player being confused if the button does anything or not. It is better to have buttons that repeat actions that are rarely used instead of a button that does nothing leaving the player wondering what its purpose. With a keyboard mapping an action to every key would be pointless and the learning curve of the game would increase exponentially with multiple key bindings.  
  
Mapping keys to appropriate buttons is essential as people have come to expect certain buttons to perform specific actions such as the left analogue stick / D-Pad should control the movement off on screen events/players with the left thumb primarily controls the movement as the thumb is flexible with a good reach alongside being good for steering and fast response. This allows for users to quickly scroll through enemies on the grid and other UI elements in the HUD. The primary index finger is used for triggers, holds, and firm button pressed which is perfect for actions such as selecting a unit or casting a specific spell.

The disadvantage of controllers in XNA for controls is that they can easily become overly complicated with combinations of button presses to perform unique actions. These can confuse the user and increase the difficulty of the game. For this reason I aim to avoid using these as much as possible and have 1 specific button for each action where possible to lower the learning curve of the game.

### Software and Hardware Configuration

The development of “Ariyana” is based around the decision to use Microsoft based Hardware and Software to make a game that is accessible on Xbox Live Indie Games alongside PC. The hardware and software configuration I have decided upon is based upon my own PC specification and a standard Xbox 360.

The software being used is XNA 4.0 and Visual Studio 2010 with C# being the program language. The hardware is a Windows PC with a Quad Core running at 3.4 GHz, 8 GB of DDR2 Ram, with an ATI Raedon 5700. In addition a standard Xbox 360 will be used to port the game onto the console and a USB Xbox controller will be used for controls on both platforms. As the game will be initially developed for PC followed by Xbox 360 afterwards the combination of XNA and C# was ideal as this allows an easy port between platforms.

### Graphics and Audio

The best graphics and audio are not always based upon pure processing power and ultra realism! Often the best graphics and audio for a specific game are those that best fit the story of the game and story by creating a specific mood and art style. An example of when this worked well is a game called Limbo. *“The game is presented in monochromatic black-and-white tones, using lighting, film grain effects and minimal ambient sounds to create an eerie atmosphere often associated with the horror genre.”* **(0)**

By creating an aesthetic for Ariyana without resorting to highly detailed three-dimensional models this allowed for a minimalistic style to be developed which will focus its attention on the game play. However the minimalistic style of the art allows for content to be created easily with limited artistic ability so from an early stage concept art and even in game art can be produced from pencil sketches and art work created in Photoshop.

Ambience will be achieved through the balance of background music and sound effects. An instrumental piece of music will be used for the background music in both the title screen and the game to create a sense of danger and foreboding to the player. Minimal sound effects will be used throughout the game as the screen will have a large amount of visual activity on screen that too many audio effects could ruin the aesthetics and ambience of the game. However the exception to this is the cut scene that will be placed at the start of the game in which it will be rely on heavy visual and audio elements to tell the story of the game to the user in a short time span.

Although content will be created for Ariyana additional content from online sources will be used and this will fall under Creative Commons 3.0 Licence. Under the Creative Commons 3.0 Licence content with this license is free to use in commercial uses of the work, free to share by creating copies, transmitting, and distributing the work, and free to use to remix the work to create an adaptation under the following conditions: The work must be attributed to the author in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work). **(2)**

Graphics

While the enthusiast gaming market is somewhat saturated with ultra realistic graphics that have been said to be indecipherable between real life and video game in 10 years there is still a market for indie games that don’t spurt top of the range graphics and instead choose to offer unique graphical styles to their games. As casual games are smaller and cheaper to develop more risks can be taken due to failure being less costly and success yielding huge profits. This has allowed developers to become inventive with styles of graphics for games and Ariyana is no different.

Ariyana is set to be a monochromatic black-and-white game that will utilise various tones and shades of black and white to emphasise the divide between good and evil in the game. This graphical style combined with minimal ambient sounds will utilise subtlety to create an atmosphere akin to that found in early film noir movies by using movies such as “Nosferatu” as inspiration for the style of the game.

When colour is no longer a factor in the graphical styling of the game it dramatically changes how the graphics for the game can be produced and the feel of the game. The artistic style used to create graphics is in the form of pencil sketches and water colour paintings that will be edited in Photoshop to add additional layers of contrast and depth to the sketches in the form of shadows and lighting effects.

These drawings will remain black and white and will be comprised of varying shades of black and white with some adjustments in the code of the game to change transparency and brightness during game play. This will allow the play on light and shadows to be presented visually in the game in addition to being an element in the story. Demons are traditionally creatures born of darkness where as humans are creatures of light and this is an element of the story I wanted to convey through the graphics.

### Graphical Style

As indie games can take higher risks than a high budget title which costs a large amount to make as there is less at risk this means indie games are open to experimental with graphical styles. There is simply no fixed graphical style that indie games conform to as recent games with graphics styles that look 10 years out of date can beat ultra realism in today’s industry. For this reason there is not one particular style that truly fit’s this genre as it is in a constant state of flux as developers are always experimenting with new ideas.

*“The thing that distinguishes one indie game from the rest is creativity, and it’s this that’s valued above all else. Indie games have to be creative because if they weren’t, no one would pay attention to them. To be unique involves taking risks, and it’s these kinds of risks indie games can afford. Something new and creative can make gaming feel fresh again.”* **(11)**

Today’s games are less about graphics and more about aesthetics as graphics are no longer the main selling point of a game. This allows small developers to make something that looks appealing with a limited set of tools and low scale graphics which can turn out to be highly successful. An example of this is Minecraft. This game was created by a single developer with graphics that look 10 years old yet it has went on to sell millions of copies and has produced its own aesthetic style proving that developers do not have to make a large budget game to be successful.

### Graphic Source

Sourcing graphics seems like a novel concept at first as the internet is filled with countless high quality graphics but the realisation quickly set’s in that almost everything you see online will have some form of copyright associated with it making it impossible to use within a game unless it falls under the creative commons license which is rarely the case. There is an array of websites that offer graphics under this license but often most are unsuitable or simply something you have to settle for and because of this reason I have considered using multiple sources to produce the graphics needed for the unique style I plan to implement in the game.

Through a combination of self designed graphics, those produced by other university students who are particularly talented at art, and suitable content from online I have managed to source the vast majority of content needed to produce Ariyana. These graphics will be used to produce various elements of the design ranging from title screens, enemies, to even including cut scenes.

All graphics created by myself or others involved in the project will be produced as hand draw sketches which will then be scanned and reproduced in Photoshop using various filters and layer masks to improve the quality thus allowing for images to be produced quickly and effectively. The benefit of producing images in this way is that each artist can spend a small amount of time producing sketches which I can then edit to improve if needed allowing for a minimum amount of time be spent on graphics yet still producing a unique style found only in Ariyana. As each artist comes from a different background such as hobbyist, to student of Games design and management and Illustration at the Duncan of Jordan, this enables each person to give unique insights into attributes I may have over looked thus improving the quality of the end product further.

In addition to this I also decided to try asking artists on the popular amateur art website “Deviant Art” for permission to use their drawings relevant to the style of Ariyana. Most artists that were active on the site were happy about their work being used and small elements of approved work from this source will be used in the game or cut scenes.



Audio

As the technical capabilities of sound hardware have increased, so has the professionalism of the people creating music and sound effects for games. The use of sound in video games has changed a lot over the years and sound has gone from being a nice addition to a necessary part of the experience in some games.

Originally sounded was there to add some background music and the odd sound effects however sound does not make a game. In much the same way that graphics don’t make a game. They are nothing more than components that go together with everything to make a game. Sound may add something, and it may be a major part of the game, but even without sound the game will still function. **(9)**

However the unfortunate thing about current thinking in game-audio, with all its allusions and aspirations to film-sound, is that it seems to have consistently neglected one essential element - that of subtlety. The dynamic peaks and troughs of film sound have such power in drawing in an audience that the impact of the louder, more spectacular sequences in a film are dramatically magnified by preceding silence. **(3)**

Well placed silence and subtlety will allow the parts of the game that need to be larger-than-life to be more powerful due to their contrast next to these areas of silence. Also the ability to wander around a virtual environment in the sections of a game where not much is happening and to simply listen to the subtle environmental sounds is extremely enjoyable and engaging to the player.

The game audience will have experienced a range of games and movies with high production values and incredibly soundtracks and trying to complete with a game of this nature is not possible for an indie game with practically no budget. However a cleverly constructed montage of silence potentially has more dramatic effect than the biggest and loudest sounds and this is the benefit of subtlety to Ariyana as a game. The structuring of how silence works in conjunction with sound is similar in a way to the film editing practices espoused by Eisenstein nearly a hundred years ago: in that expressive power is only gained when these elements are edited together and deliberately played against one another. These techniques can be clearly seen in the horror genre of films and most notably classic noir horror movies in the era where sound was a recent addition to horror movies. **(8)**

### In-Game Audio

Ariyana will use used dark, chilling and atmospheric sounds to create a sense of tension and fear for the player. This will be achieved through the use of subtle use of background music during key stages in the game play and sound effects that will be used sparingly to create an atmosphere with highs and lows that heightens the sense of foreboding danger during pivotal portions of game play and silence during portions of game play where nothing is happening.

The background music will be used in the title screen and during game play at certain key moments. OpenGameArt.Org will be the source of this audio as the production time for a track of equal quality would be beyond what is feasible for the project. As the audio acquired from this source is licensed under the creative commons license it is free to use in this project without fear of breaching any copyright protection.

For sound effects there are a number of (expensive) sound-effect libraries, often produced by movie companies such as Warner and 20th Century Fox however in keeping with the style of the game and creating an engaging audio experience I have decided to involve a sound engineer to aid in recording some simple sound effects. **(5)**

### Source of Audio

The sources of sound are not always as obvious as they may seem. There are the very obvious sources of sounds, such as recording the sound an object makes in a certain situation e.g. the sound of a sword clashing with a shield. Realistically procuring a sword and shield and lunging at your friend may produce realistic sounds but it’s far from practical. An alternative is sound libraries as they offer a massive collection of sounds but this comes at a price and in this project it is not feasible to use a sound library due to the cost associated with purchasing one.

Producing sound effects for Ariyana will involve a sound engineer from “Creative Sound Production” at Abertay University who will aid in the production and editing of effects by using techniques such as Foley art.

*“Foley art is a method in which sound effects are achieved by “faking” the noise. Simple sound effects such as walking on gravel can be achieved by scrunching up a crisp packet and recording the sound then distorting it a bit in sound editing software to create what is seemingly the perfect noise.” (9)*

As XNA is limited to wav, mp3, or wma files even though it converts the file type upon building the project producing the sound effects all sounds effects will be recorded in the required formats .Conversion still occurs when a custom content loader is created to load other file types so the format of audio is irrelevant as long as it is accepted by the program and a high quality recording. The benefit of recording sound effects is it ensures a suitable format which allows files to be imported into XNA and removes limitations imposed by alternatives such as free sound effects available online that may be incompatible file type, lower quality, or unsuitable for the game.

Sound editing software (Audacity) will be used to edit sound effects and even combine effects to create much more impressive audio sequences in the game. Free programs often have more limited features than paid professional software however they are suitable indie developers. As Audacity is a free digital audio editor and recording application i have decided to use it as it allows for audio recordings from multiple sources and it can be used for post-processing of all types of audio by adding effects such as normalization, trimming, and fading in and out. **(7)**

Cinematic Cut Scene

Games are a participatory form of art. Players do not passively experience games as an audience member experiences film. If one does not actively play, the plot does not progress. Most games have a very linear plot in which the developer seeks tell a set story. **(6)** Ariyana will uses cut scenes to progress a linear story instead of spending large amounts of time trying to make a larger game with an interactive story during game play. These cut scenes will be deployed at the start and end of the game to provide an entry point and exit to the game. These will be created from a combination of both graphics and audio created for other elements of the game alongside animations added in post production to give the player a more realistic cinematic experience offered by other games.

Ariyana’s cut scene’s should be solely focused on making players feel a sense of danger, thrill, and excitement during in which they feel actively engage in the story during game play. By viewing the cut scene it effectively allows the user to quickly gain an insight into the game world without the need to read long paragraphs of story or spend any considerable about of time learning what is happening within the game. As the saying goes “A picture is worth a thousand words” and within a few seconds the story can be told with other visual aids to make it a cinematic experience and by making the cinematic compatible with the game to utilize the participatory strength of the medium it can enhance the game in multiple ways.

To progress the plot forward the user must play the game to defend the castle from the horde of demons and only upon the start and end of the game will cut scenes be used to tell the story that has unfolded. The player will watch passively as the events in the cut scene unfold but it allows for a game to include an ending that fits with a story line and provides a sense of completion instead of a gang that leaves the user hanging once complete with nothing to do but exit the game. Engaging the user in multiple ways and actively progressing the story allows for a much richer story to be told that the user will care about and will come back to play more of.

Only games allow for another person to advance the narrative of a story. Cut scenes undermine the most important trait of playing a game. They force the player to become a spectator rather than a participator. They cheaply employ film techniques to express a story rather than challenging the player to not only experiences the world, but to react to it on a constant basis. Developers need to stop trying to make games be films with cut scenes that separate game play from story. They need to further the medium out of its infancy and into maturity by taking advantage of its strengths. They need to make people experience by doing and not by watching.

**Game specific features**

The game is required to have a minimum of one game specific feature from a list of features however Ariyana will use 2 of these heavily and elements of another two to a degree in the game. The following is game specific features that will be added to the game.

* Real time game play
* Collision detection
* Artificial intelligence

**Real Time Game play**

Traditionally chess is played as a turn based game however the inspiration for this game came from traditional chess and wizard’s chess from the universe of Harry Potter. In wizards chess the game is still played in a turn based style however the pieces come to life and perform actions such as destroying other pieces or dragging them off the board. This element of wizard’s chess appealed to me and fitting with the more traditional style of tower defence game a real time game provided the fast action game experience I wanted for Ariyana.

The real time game play element is crucial towards the game mechanics and relied upon heavily by the game to provide the appropriate player experience. The reason for selecting XNA to develop Ariyana was its suitable for developing real time games.

Within the game all elements of game play will be happening in real time. Enemies on the battlefield will move towards the castle in real time and collision detection will be carried out whenever an enemy is about to intersect a new square on the grid. If a collision is detected then the enemy will change states from moving to fighting and its behaviour will change. It will no longer attempt to move forward and instead will attack the enemy in front of it until it perishes and the enemy can return to a moving state. Instead of changing state betweens turns the enemies on screen will continuingly move forward and attack units in their path. To determine which lane of the board the enemy will follow a form of AI is needed to determine which lane is the most viable option which must be calculated in real time whilst the enemy is in the spawning state. Within a few seconds the enemy must be able to spawn, move, and attack a unit. To accomplish having multiple enemies on screen alongside units the game will rely on states that can change multiple times within a second to allow behaviour to be modified in real time.

Using states allows the phases in the game to have their own set of rules without the need to check conditions multiple times a second. Actions can be performed on screen when a state changes and the game can be updated in real time and displayed back to the player so actions such as spells and in game events are only trigged when pre conditions are met such as a button press or the game time has reached a certain point.

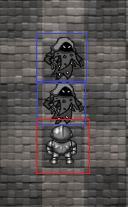
As everything is happening in real time enemies and units on the board need to be aware of each other and know how to react to events in real time. Such as if an enemy is behind another enemy and its range is too short to change to a fighting state then it changes to waiting until either the enemy in front dies or moves forward. This allows for multiple enemies in a lane to be aware of each other and react according to any changes in real time as if no change is made in the first enemy and the second enemy cannot do anything then the cycle resumes itself until a time in which the state changes.

If every enemy and unit on the board were to behave independently then each state would need to be checked every update cycle and anytime an enemy or unit made a change on the board it would reflect upon all others. Whereas when enemies and units have states they can simply be ignored until their state changes. In effect this allows the enemy or unit to continue doing what they are doing until something changes in the system resulting units behaving correctly as all units and enemies on the board should be able to move and attack at the same time without conflicting with each other.

**Collision Detection**

Collision detection in the game will be confined to checking if enemy collision boxes intersect with the collision boxes other enemies or units on the battlefield. In addition collision detection will be used for the relocation of units on the battlefield to ensure they are placed on a square that does not contain any other unit or enemy.

The collision check will be performed whenever a player attempts to move a unit and every time an enemy attempts to move forward 1 square. The collision detection for each enemy only needs to check the square it is attempting to move into to see if it is either empty, has a unit, or is the castle. This allows for minimum checks to be done as at any time every enemy on the screen will be only performing 1 collision check on the grid to see if I can move forward.

The collision box used for enemies will be smaller than the collision box used for units as the reason for this is that it gives units on the battlefield the opportunity to attack first which is often needed to balance the game as no player charachters have the advantage of absolute precision where as players do not and small advantages such as a larger collision box to attack enemies allows the player a chance to see the unit react to the presence of the sprite attacking it and display the animation. Alternatively if the collision box was the same size it would appear both enemies were attacking at the same time or possibly missing each other entirely. The collision box the enemy sees on the unit is smaller however the units collision box is larger to detect enemies before enemies have a chance to attack the unit.

Collision detection will be used to change the state of units and enemies as if a enemy collides with another enemy then it enters a waiting state where as if an enemy collides with a unit then it enters a fighting state. Which can be seen in the two images above. The first image shows the enemy attacking the unit where as the second shows an enemy waiting behind the attacking enemy.

**Artificial Intelligence**

Different enemies will use different route finding algorithms that will be used in their spawning phase to determine which path way they will select in the game. Once an enemy selects a lane on the grid they cannot change lanes so they are fixed on that path so the algorithm is only used to determine which lane is the most viable option. The rest of their behaviour will be generated in their movement and fighting behaviour that is established when created.

The algorithms will use different factors for different enemies based upon who difficult the enemy should be to defeat and how intelligent it should be. Such as enemies at the start of the game will find the shortest path based upon how many units are currently occupying that lane. Whereas more advanced enemies can differentiate between the types of units and calculate the cost of the lane based upon their defence, attack, or a combination of the two allowing intelligent enemies to pick the weakest lane and not necessarily the shortest.

This allows enemies to behave in different ways but with a slightly predictable pattern that the player will learn thought playing the game. To the player it will seem as if there is an element of intelligence to the selections of pathways and will appear as if the enemies are working together to beat the player by selecting different routes.

The algorithms used to determine which lane to select will be calculated every time a unit is spawned in the game. The advantage of only performing this calculation during spawn time allows one calculate to be used for placing all enemies as whenever an enemy is placed in a lane the cost of using that lane is increased to an infinite state making it impossible to use until the enemy progresses 1 square forward. In effect if all lanes have just spawned an enemy then all enemies will be added to a queue that will have to wait until the next update cycle when the square is empty to spawn. This helps to balance the demand of the game as during any update cycle no more than an x number of enemies equal to the breadth of the board can spawn in the game.

**Game Code Structure**

The structure of Ariyana will consist of MVC and 4 design patterns (Strategy, State, Singleton, and Observer).

**MVC design pattern**

Initially when I considered the design of Ariyana the MVC design pattern was not at the forefront of my mind and as XNA already implements its own version of MVC already I thought it would be easy to create a simple version of my game. After some thought I realised that to code the game it would require everything to be working before the game would work and any changes would either cause the game to become riddled with bugs or impossible to manage when any new features are added.

*“Games are simulations that have a life of their own. Without player input, they’ll happily send some horrific creature over to start pounding on your character’s skull. That will probably motivate a few button presses. The system that controls this ongoing activity is the main loop, and it has three major components: grabbing and queuing player input, ticking the game logic, and presenting the game state to all the game views, which means rendering the screen and playing sounds****”(*22)**

There is where the MVC design pattern comes to the solution. Effectively separating the model, view, and controller in the game allowed for the code to be optimised for extendibility allowing a simple version of the game to be created and improved. By having these three separate layers it also increases reusability and encapsulation allowing for portions of the game to be coded, made to work, and remain that way regardless of any other changes to the game solving part of the initial problem of changes in the code causing disastrous effects that ripple throughout the rest of the game. Additionally when the game is completed it will allow the game to be easily ported to the Xbox 360 with minimum changes required. As the model, view, and controller will all be separate.

The game will be real time and will be heavily event driven. Triggers in game will be activated to start and end events as the game will be running in real time so at any second the game state can change and must be reflected so on screen as fast as possible. Changes in the game state such as moving from the menu system to the game will trigger a range of events such as input from the user from a device such as a mouse, keyboard, or gamepad. Once any input is detected the system must trigger an event from this external input and perform actions within the game.

When the game logic makes changes to the game state, such as creating or moving an enemy, a number of game systems will respond to handle the event automatically whenever game play starts. An example of this is once a timer has incremented past a set amount of predetermined time then the game must change its current phase to the next phase in the game that will contain its own set of rules of how the game will progress. Even without input from the user the game will continue to run and trigger events. It is these events that will require the player to respond as an enemy on screen progressing towards the castle will require the player to intervene.

**Model (Application) Layer**

Input is essential to any video game even if it is a single button press that triggers an event. There is a range of input devices however the three that will be used within Ariyana are the keyboard, mouse, and gamepad. The framework of XNA handles the calls to the operating system and device drivers so the only coding needing is to code the key bindings and read the input and trigger the event associated with the key binding.

The way in which MVC handles input differs from how I originally planned due to the separation of the model from the controller. Of these three the only device that will have a model class is the gamepad as this will contain the configuration for the key bindings for that gamepad allowing it to be changed by the player if they feel it is too difficult to use. The keyboard and mouse are built into the framework of XNA and will only be used on PC. Later these controllers can be removed when they are no longer needed. The state of these input devices will be translated into game commands. This is done by the Input manager handling all input and calling upon the game board to perform the request. These requests are sent between classes in the controller layer that handles the logic and based upon the event the action will either be performed in the game logic such as “Heal Unit” or if it is an event that triggers a game view it may be to show the “unit menu” to select from.

The use of the gamepad class allows for a configurable gamepad that players expect of modern games. Although there are standard controls for certain games some people like to customise their controls for what personally works better for them. By creating a configuration file the key bindings can be mapped to the functions allowing the user to select which button triggers which function if they choose to deviate from the standard configuration.

Controller (Logic) Layer

The game logic layer is the glue that holds the entire game together. It defines everything that exists in the game universe and how objects interact in that universe. The game state can be changed by internal and external stimuli such as events triggering state changes in game. Input from the user causing changes in the game state such as the user selecting “new game” from the menu system. The vast majority of state changes in the game will come from AI processes that are taking action within the game to kill players units and win the game by breaching the castle.

The controller layer in MVC completely separates the machine the game is running on to how the game is presented to the player which is essential when porting to the Xbox as visual changes may be required in order to accommodate a larger screen and various screen resolutions when played on a console. This ensures that the controller (logic layer) only requires the minimum amount of changes to port to a new device.

The game logic will contain the data that describes the game board and the units/enemy’s within the board alongside all the properties of each. For the game board, the data describes what each piece of the board contains. The enemies and units will contain data such as health, attack power, defence, special ability’s. The game board will be in charge of deciding what happens to the units within game pieces and how they respond to changes in state such as from moving to colliding with an obstacle then entering a fighting state.

The game logic only cares about what state it is currently in and what state it will progress to next if the game has not ended. The same is used for enemies on the game board as they only need to know where they are and where they are going. Any state changes in-between are to help enemies progress forward past obstacles in the game by changing state to surpass the obstacle. The game will progress without input from the user however this game is likely to end very quickly as once an enemy beaches the castle the game is over. Input from the user will affect the current state of the game and trigger in game events and actions. Outputs from the logic layer will be any object on the game board having their properties updated such as position, health, and current state.

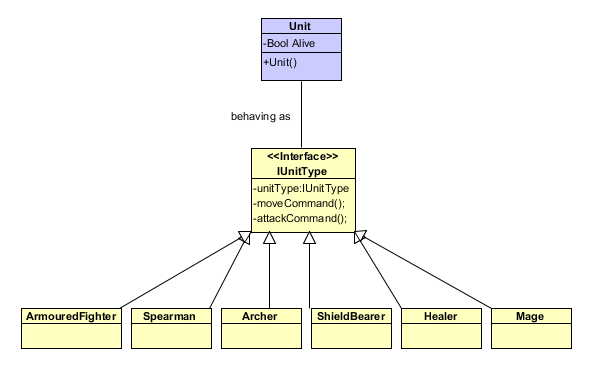
View Layer

The game view layer is a collection of systems that communicates with the game logic to present the game to a particular kind of observer which in this case is the Player. The view layer will communicate with the logic layer to handle all of the rendering of user interfaces, sprites, effects, audio, and video. In Ariyana the view layer is used for displaying the game to the user as the logic handles which state the AI is in and the change of state determines how the AI reacts. Unlike an AI agent making decisions the AI within Ariyana is predefined and will only have some elements of random behaviour associated with it. Events and state changes made by the controller are sent to the view and the view will display these changes to the player on screen by rendering any UI and Sprite to be updated.

The game view for the player often has a lot to do to present the view of the game state as this can update multiple times per seconds in a real time games. The view will handle rendering the sprites on screen from a specific top down view, it will send audio to the speakers, spawn particle effects, and possibly provide force feedback through the control if there is enough time to implement this feature. The view can have different implementations and as many views as required can be attached to the game. Such as in my UML class diagram the game screen has multiple implementations of what the user will see when a state is changed in the game such as the user selecting to pause. These views all inherit from a game screen view but what the user sees in each is slightly different such as in the game screen enemies will be moving whereas when paused the game logic will pause too and the sprites on screen will remain still and a partially transparent layer and message box will be displayed to show the game is paused.

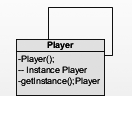
**Strategy Design Pattern**

The strategy design pattern splits the behaviour of a class from the class itself. The advantages of this is that it fits into the MVC model being used as the model and controller (Game logic) can be separated allowing one model to have multiple implementations of the same object with different behaviours.

The strategy class will assign unit and enemy behaviour allowing both to conform to the behaviour specified for that type of unit. This allows for extension by adding new types of units whilst keeping the existing classes closed to modification and encapsulates the behaviour of units. The advantage of this design pattern is that it allows units to be added incrementally and any changes to the code won’t affect any other concrete strategies so unit and enemy types will be maintainable regardless if there is 1 or 100 as each can function independently without affecting any others.

* **Context: Unit :**It stores context information such as current state, position, attack range, relatively to other units and enemies, etc, and passes necessary information to the Unit type (Strategy) class.
* **Strategy: IUnitType:** is an interface that defines the behaviour of a unit by assigning it a type of behaviour.
* **Concrete Strategies: Armoured Fighter, Spearman, Archer, Shield Bearer, Healer, Mage:** Each defines a unit specific behaviour that specifies how an enemy moves or attacks by using information such as current state, position, attack range, unit health, and even if the unit is still alive.

**Singleton Design Pattern**

Singleton is one of the simplest design patterns and uses only one class which is responsible for instantiating itself to make sure no other class can create a copy of it. To do this the constructor is set to private making it impossible for outside classes to access the constructor to create a new instance. Sometimes it's important to have a single instance of a class. In Ariyana singleton is used for the player as there should only ever be 1 player in game at anytime as it is a single player game.

Singleton ensures that only one instance of the class is created and that this instance is accessible globally by all other classes in the program. The benefit of doing this is that the player class provides a global point of access to that instance which will be needed by various classes within the game for different purposes. This effectively limits all other classes in the game to use the same instance and resolves issues that may have occurred with multiple instances of the player class.

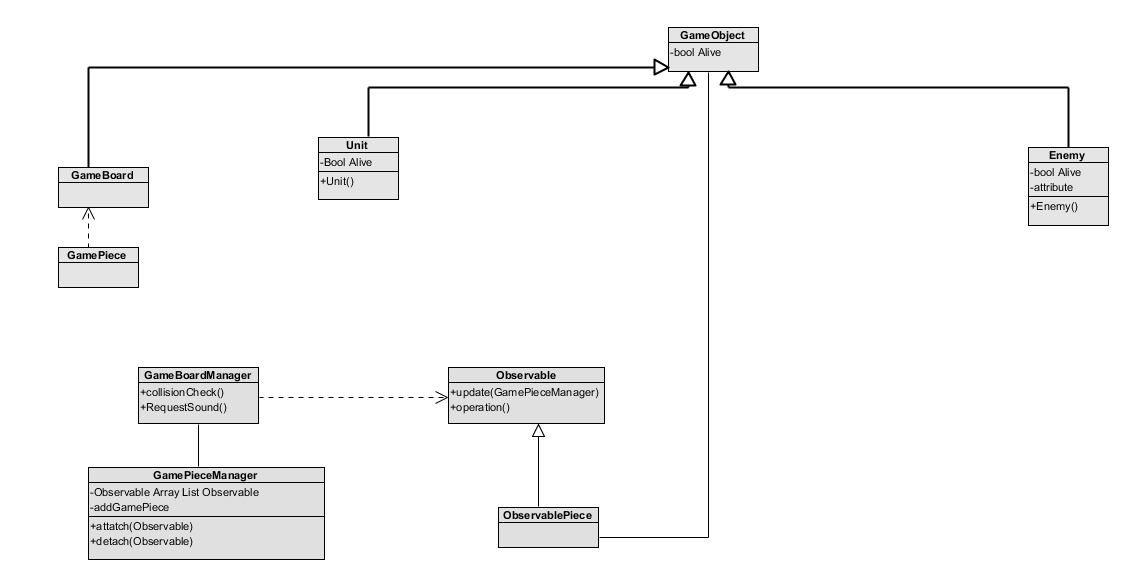
**State Design Pattern**

State is very similar to the Strategy design pattern however the main difference between the two is state makes the behaviour change at runtime by making the behaviour dependant on the current state where as Strategy provides fixed behaviour for the object that it should follow. State allows an object to alter its behaviour when it’s internal state changes.

In Ariyana the state design pattern has been used for controlling how the game state changes depending on what state the game is currently in and allows the game to function differently in each state. The following example shows the game states in which the game can exist in only 1 at any given time. Each state will change what happens in the game such as the first state will control the menu system, the second will load all of the content needed for the game and initialise any resources or classes needed by the game, the cinematic will display the opening and closing cinematic to the player, playing will control the game flow when the game begins such as each of the three phases and the events that occur during those phases, and paused will cease all functions within the game until the player selects to resume the game.

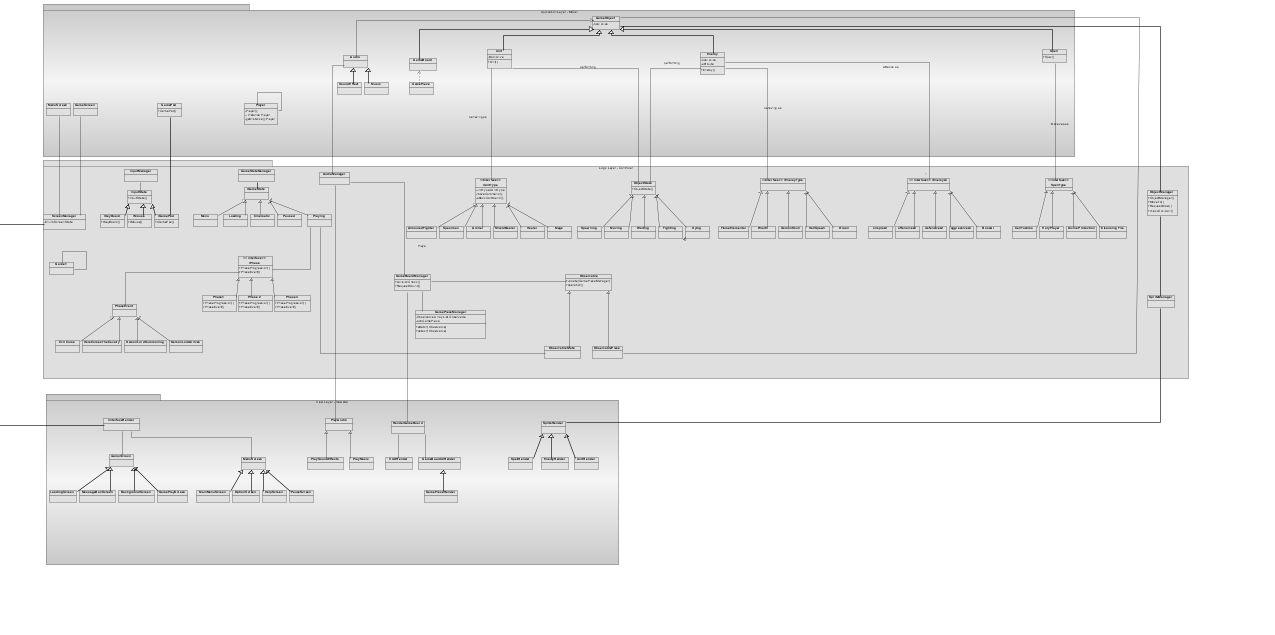
* **Context ( Game State Manager ) :** Stores and maintains an instance of a concrete state that is used to define the current state. The game state manager will only contain 1 instance of the game state as the game can only ever be in 1 state at any given time. GameStateManager will be a high level entry point to the controller layer of the MVC model.
* **State (IGameState) :**Defines an interface for encapsulating the behaviour associated with a particular state of the Game State Manager
* **Concrete State (Menu, Loading, Cinematic, Playing, Paused):** Each Concrete state implement a behaviour associated with a state of Game State Manager (Context). Each concrete state will have a behaviour that will be used to control the game through each state and onto the next.

**Observer Design Pattern**

Observer defines a one to many dependencies between enemies and units on the game board so that when a unit or enemy changes state, all its dependents are notified and updated automatically. Observer offers the ability to change the state in one object that must be reflected in another object without keeping the objects tight coupled. This is useful when enemies or units die as they are not tightly coupled to the game board. The benefit of this design pattern is that it allows all enemies and units on the board to be aware of others in their lane alongside all others on the board so their states can change and appropriate behaviour be assigned to each enemy and unit on the board.

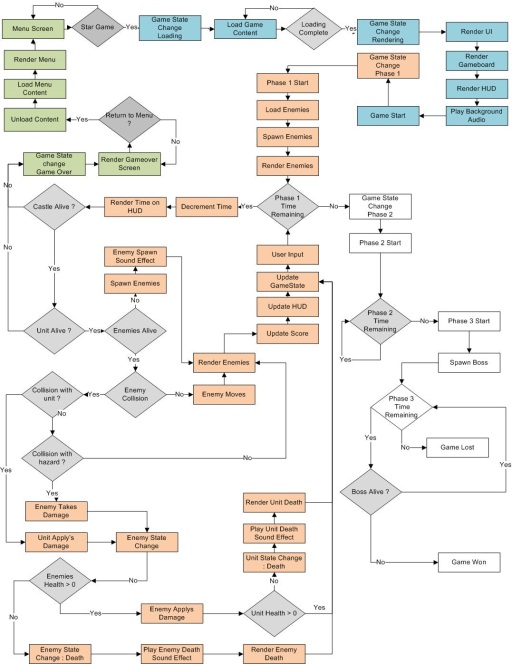
* **Subject (Game Board Manager):** The observer logic is implemented in the game board manager. It knows it is the observer and has any number of observers but in this example of the class diagram in Ariyana it is only 1. It provides an interface to add and remove observer objects at runtime and it keeps a list of all it observables and informs them about the latest game pieces.
* **Observer (Observable)**: provides an update interface to receive signal from the game board manager.
* **Concrete Subject (Game Piece Manager):** The game piece manager stores the subject state of interest by the observer and sends notifications to the observer.
* **Concrete Observer (Observable Piece):** The observable piece maintains a reference to the concrete subject (game piece manager and observer state . Also implements an update method to update this.

**UML Class Diagram**

The UML class diagram seen below and in the appendix (Figure 1) will display all of the classes used in Ariyana and demonstrating the MVC design pattern along with the additional design patterns already mentioned. In the diagram the top layer is the model, the middle layer is the controller, and the bottom layer is the view. The view and the model never directly connect with each other and instead use controller classes (the glue) to link the two together.

One of the first problems that I had to address when creating the UML class diagram for Ariyana was that initially everything started off in a hierarchy however once MVC was implemented this was greatly reduced to a version that was close to the finished version. With knowledge of additional design patterns some classes were removed and the structure was greatly improved. Now all the classes fit easily into the MVC model whereas before classes were behaving as the model, view, and controller. This change allows for reuse, extension, and encapsulation which were all initially missing in the first version. The design patterns that were used for enemy/unit behaviour structured the code in such a way that is closed to modification whilst remaining open to extension. This allows for functionality to be added and an initial version of the game created which can be built upon instead of having to code everything from the start and later modifying it whenever a change is made. Code is now open to reuse and future extensions.

**Behavioural Diagram**

The behavioural diagram was a useful aid in mapping out how game mechanics and other game components would work together to produce the final game. Using flow charts I mapped the structure of how the game should run and this helped to identify issues I had over looked and by working backwards it identified what other components were necessary in the diagram to use a feature I had already included.

The diagram was broken down into 4 seconds. The first section was for the menu system that would change the initial state of the game to loading then playing to start a new game. When this occurs any content that is needed is loaded into the game and any classes or objects that will be needed at the start of the game are initialised.

The additional three sections are devoted to the three different phases of game play. During these three phases a large portion of the game changes and instead of attempting to integrate this into the existing flow chart I decided to separate it and include states in the game so when the state changes the phase changes and everything in the game will change with it. This helped clarify the UML diagram by identify additional classes that we needed.

**Time management plan**

Using MVC and other design patterns has allowed the code of the game to be broken down into manageable chunks in which functionality is encapsulated to a few classes that once complete will be closed to modification and open to extension. The benefit of this towards the time management plan of developing Ariyana is that is allows a small portion of the code to be written and tested by creating a prototype effectively creating a playable version of the game from an early stage which can then later be expanded upon to add the rest of the content thus allowing the time management to effectively become task management.

The first task may involve creating a working prototype with reduced functionality incorporation various classes from each of the three layers of the MVC model. This gives a better estimation of when the task will be complete instead of the overall time it will take to implement everything. From previous experience on large projects it is clear it is hard to estimate everything when you are still learning how to carry out the task. Unexpected problems occur and estimations of time needed often change due to unforeseen circumstances.

Setting tasks such as completing a working prototype version by the end of week 1 allows for tasks to be broken down into manageable chunks whilst also addressing the issue of unforeseen problems. If a problem does arise the tasks can be adjusted to be completed in the time by producing part of it to create a working version instead of implementing the entirety of the code before checking if it works as intended.

An example of this would be writing the unit and enemy classes for the ten individual characters implementing all their behaviour before being able to test everything works and creating a playable version of the game. Instead by focusing on producing the minimum such as 1 unit and 1 enemy character the design pattern leave the code open to extension at a later time whilst allowing more time to be spent on other areas of the code to produce a playable version sooner rather than later. This contingency time allows for time to be spent on issues that arise if something does go wrong and in the event it doesn’t then that time can be spent on implementing the additional characters and behaviours.

In this project I will be using a Gantt chart for my time management as it is flexible and adjusts based on how long has been spent on each task and the time remaining before the submission deadline. The advantage of this is that it will visually show if a task is running over time and how the plan has deviated from the base line from the start of the project. This addresses any issues of the game not being completed on time by continuingly monitoring and adjusting the work being completed. If a task does take too long to implement something then sacrifices can be made elsewhere in the game to reduce additional components.

Instead of forcing tasks to be completed in a specific order they will be arranged into an order that they will most likely be required in order to produce a prototype and any changes or deviations from this order will be adjusted in the Gantt chart to refractor the time and effort spent on each task. In addition tasks may not be fully complete until the ends of the project however they will progress through incremental stages in which what are coded is complete and only extensions to the program are needed.

### References

**Design References**

**Cut scene art work** (Figure 1,2,3,) - Kyle Harrison(September, 2012) Ariyana cut scene artwork

**Enemy Sketches** (Figure 4,5,6) - Callum Stewart (2010) Designed for Ariyana

**Unit Sketches/Spear Maiden**

HisokaKatsu. (2010). *Sketch Drawings.* Available: http://hisokakatsu.deviantart.com/art/Warrior-Sketch-111163284. Last accessed October 4th 2012.

**Spell and Event Icons**

josepzin. (2012). *Magic and mechanic in-game interface icons.* Available: [http://opengameart.org/content/magic-and-mechanic-in-game-interface-icons. Last accessed September 2012](http://opengameart.org/content/magic-and-mechanic-in-game-interface-icons.%20Last%20accessed%20September%202012).

**Hardware Software References**

**(1) Making Indie Games**

Neil Long. (03:35pm September 4 2012). *Making Xbox Live Indie Games.* Available: http://www.edge-online.com/features/making-xbox-live-indie-games/. Last accessed October 1 2012 .

**(2) Why use XNA**

Doolwind. (January 7, 2008 ). *Why You Should Use XNA.* Available: http://www.doolwind.com/blog/why-you-should-use-xna/. Last accessed October 1 2012.

**Graphics and Audio References**

(0) **Limbo** Wikipedia. (2012). *Limbi.* Available: http://en.wikipedia.org/wiki/Limbo\_%28video\_game%29. Last accessed 2nd October 2012.

(1) **The Computer Games Design Course : Music and Audio**

Jim Thompson. (2007). Music and Audio. In: Thames & Husdon *The Computer Game Design Course : Principles,Practices and techniques for the aspiring game designer*. 2nd ed. London: Thames & Husdon. 160-164.

(2) **Creative Commons Licence**CreativeCommons. (2007). *Attribution 3.0 Unported (CC BY 3.0) .* Available: http://creativecommons.org/licenses/by/3.0/. Last accessed 2nd October 2012.

(3) **The Computer Games Design Course : Cut Scenes** Jim Thompson. (2007). Cut Scenes. In: Thames & Husdon *The Computer Game Design Course : Principles,Practices and techniques for the aspiring game designer*. 2nd ed. London: Thames & Husdon. 60-61.

(4) **The Computer Games Design Course : Gathering Materials**

Jim Thompson. (2007). Gathering Materials. In: Thames & Husdon *The Computer Game Design Course : Principles,Practices and techniques for the aspiring game designer*. 2nd ed. London: Thames & Husdon. 88-91.

(5) **Cut-scenes: Developers Playing the Movie Game**

Stephen . (Jul 15, 2012). *Cut-scenes: Developers Playing the Movie Game.* Available: http://www.1up.com/do/blogEntry?bId=9111601. Last accessed 2nd October 2012.

(6) **Audacity**Wikipedia. (19 September 2012). *Audacity (audio editor).* Available: http://en.wikipedia.org/wiki/Audacity\_%28audio\_editor%29. Last accessed 2nd October 2012.

(8) **Subtlety and Silence**Rob Bridgett. (3rd October 2012). *Subtlety and Silence.* Available: http://www.zero-g.co.uk/index.cfm?articleid=722. Last accessed 3 October 2012.

(9) **Use of Sound in Video Games**Sean Shortreed. (2011). *Use of Sound in Video Games.* Available: http://thisismyblogandyouwilllikeit.typepad.com/blog/2011/03/use-of-sound-in-video-games.html. Last accessed 3 October 2012.

(11) **Why Indie Games are the future**Jeff Han. (2012). *Why indie games are the future.* Available: http://www.wearehappylandfill.com/why-indie-games-are-the-future/. Last accessed 3 October 2012.

**Other Graphics**

**(22) The Main Loop**

McShaffry M.. (2003). The Main Loop. In: Paraglyph Press *Game Coding Complete*. : Paraglyph Press. .

**(23) Sprite Sheet**

wulax. (2012). *Medieval fantasy character sprites .* Available: http://opengameart.org/content/lpc-medieval-fantasy-character-sprites. Last accessed 27th September 2012.

**Wraith ( Figure 14)**

Jesse McCarthy. (2012). *Grue.* Available: http://opengameart.org/content/grue. Last accessed September 2012.

Appendix

**Figure 1**



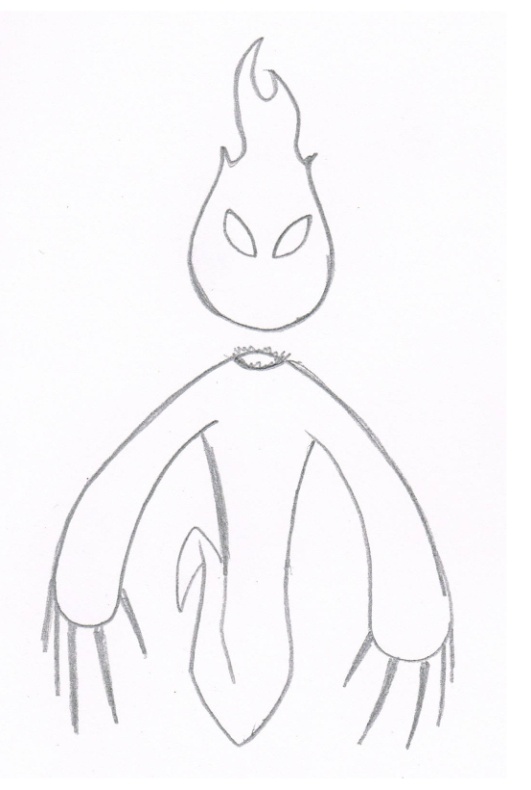
**Figure 2**

**Figure 3**



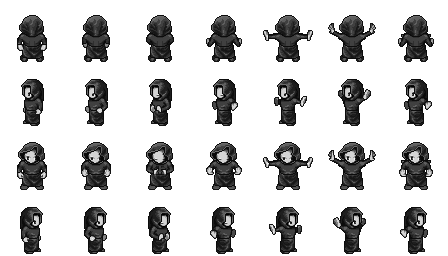
**Figure 4**  **Figure 5**

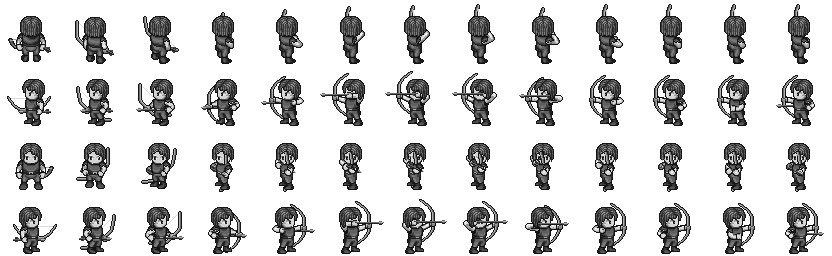
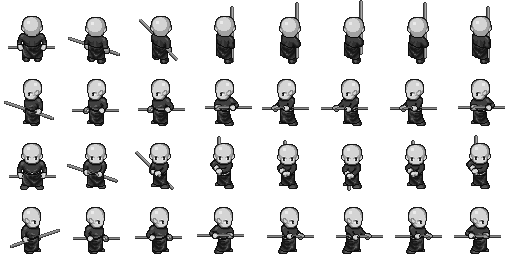
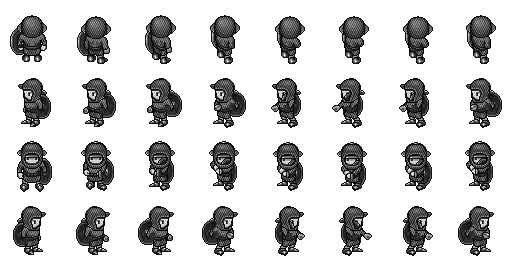
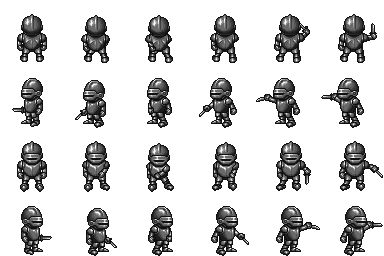


**Figure 6 Figure 7**

****

**Figure 8-13 SpriteSheets**

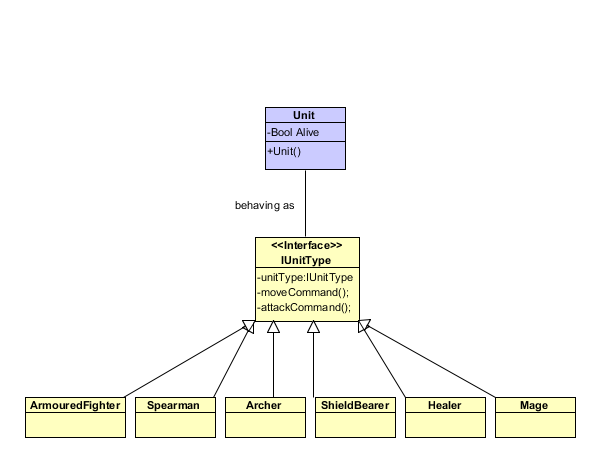




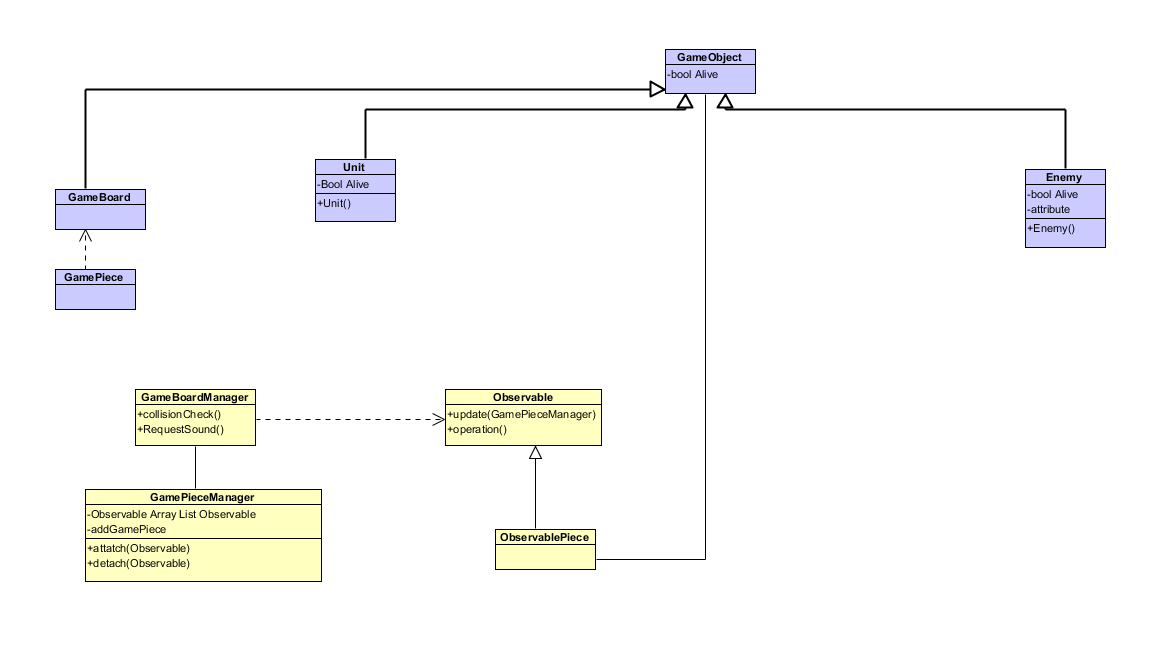
**Figure 14**



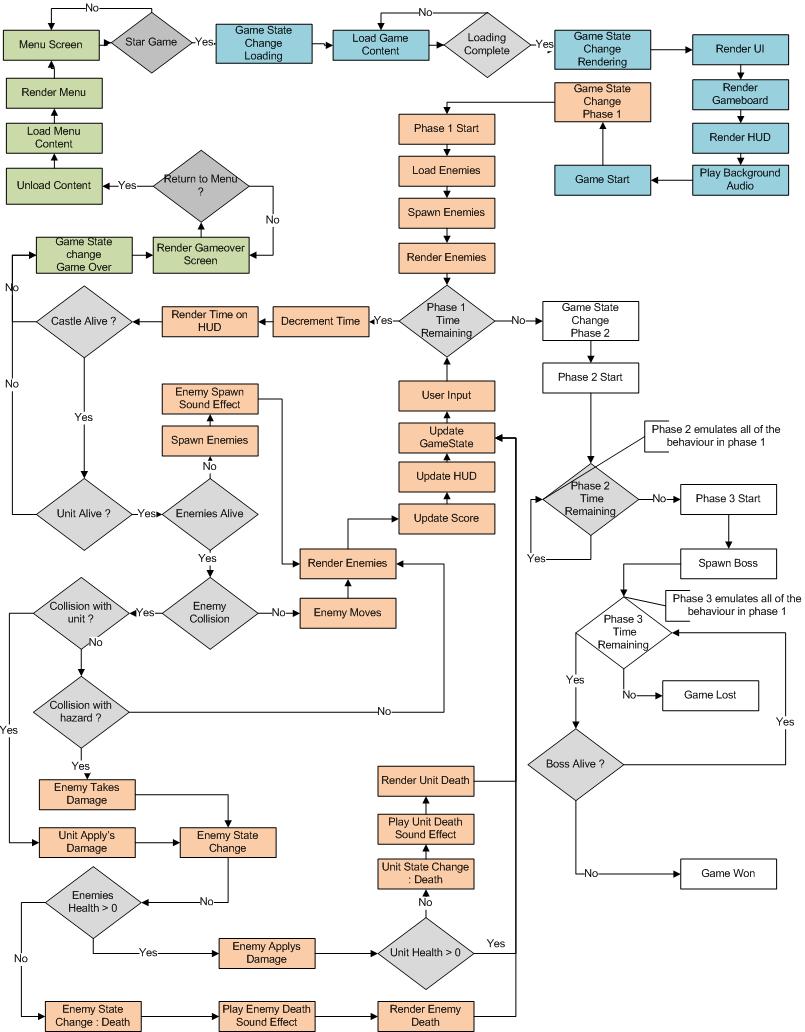
**Strategy Pattern**



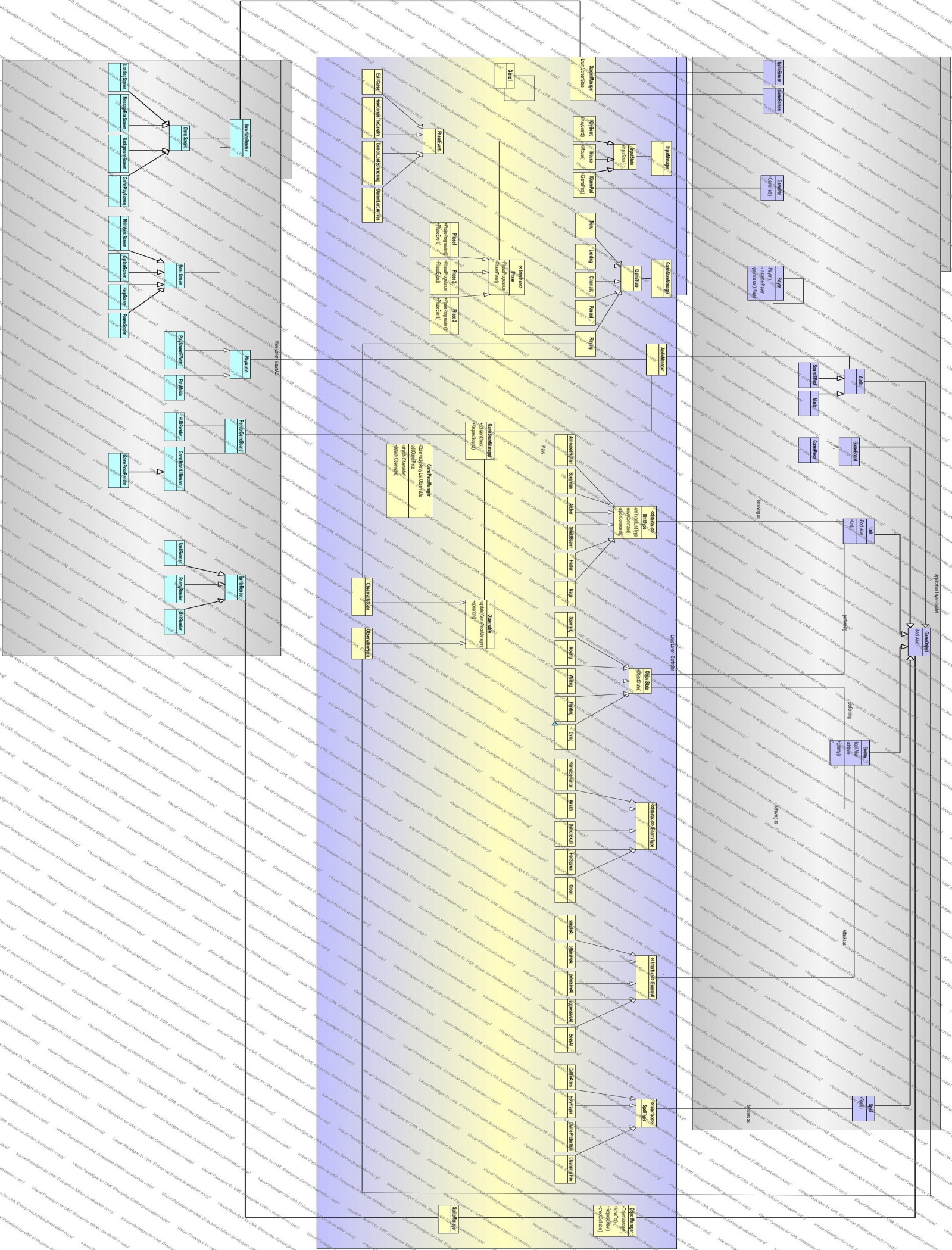
**Observer Pattern**

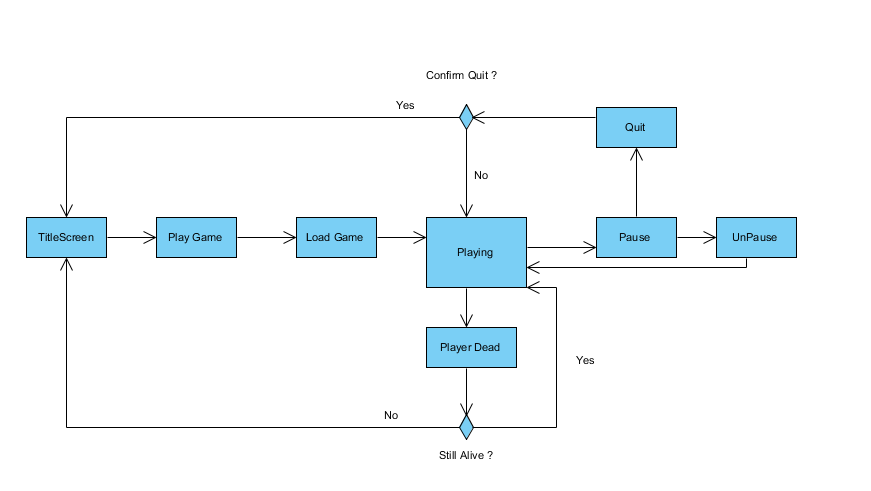


**Game Flow Diagram**

****

**Class Diagram**



**Main Menu Flow Diagram**

**Feedback**

**Game code Structure**

In the game code structure I am unsure if the observer design pattern has been used correctly.

**UML Class Diagrams**

In the UML class diagrams I wasn’t sure if the strategy pattern for enemies and units would hold the interface and concrete context in the logic or if it would be part of the model. I assumed as it was performing a function it would be part of the logic as it determines how enemies and units move.