**Cannon in C#**

Fight cannons... with cannons!



COMP 376 : Introduction to Game Development

Team 6: Phoenix Down Studios

Roxanne Beriault (9220356)

Adamo Maiorano (9336710)

Phuong-Anh-Vu Lai(5644399)

Jonathan Casquinha (9228284)

Fall 2011 - Revision 2.5

Table of Contents

[1. Executive Summary 3](#_Toc310978497)

[2. Overview 5](#_Toc310978498)

[3. Related Games 6](#_Toc310978499)

[3.1. Rampart 6](#_Toc310978500)

[3.2. NetStorm 8](#_Toc310978501)

[3.3. Worms 9](#_Toc310978502)

[3.4. Tetris 10](#_Toc310978503)

[3.5. Monkey Island 11](#_Toc310978504)

[4. Player Composite 12](#_Toc310978505)

[4.1. Casual Gamer Profile 12](#_Toc310978506)

[4.2. Transitional Gamer Profile 12](#_Toc310978507)

[4.3. Hardcore Gamer Profile 13](#_Toc310978508)

[5. World 13](#_Toc310978509)

[5.1. Setting 13](#_Toc310978510)

[5.2. Backstory 13](#_Toc310978511)

[Premise for gameplay 14](#_Toc310978512)

[6. Characters 14](#_Toc310978513)

[Sir C. Sparkington 15](#_Toc310978514)

[Cannons 15](#_Toc310978515)

[7. Progression Graph 16](#_Toc310978516)

[8. Art Direction 17](#_Toc310978517)

[8.1. Visual Language 17](#_Toc310978518)

[Shape and Silhouette 17](#_Toc310978519)

[Proportions 18](#_Toc310978520)

[Orientation 19](#_Toc310978521)

[Color Palette 19](#_Toc310978522)

[Motion 20](#_Toc310978523)

[Depth 20](#_Toc310978524)

[8.2. Reference Art 21](#_Toc310978525)

[8.3. Concept Art 21](#_Toc310978526)

[Cannons 21](#_Toc310978527)

[Towers 21](#_Toc310978528)

[Font & Text 22](#_Toc310978529)

[9. User Interface Storyboards 22](#_Toc310978530)

[10. Tags and Dialogue 25](#_Toc310978531)

[11. Technology Plan 27](#_Toc310978532)

[12. Software Architecture 28](#_Toc310978533)

[13. Controls 30](#_Toc310978534)

[Main/Pause Menu 30](#_Toc310978535)

[Fortify Phase 31](#_Toc310978536)

[Cannons Phase 31](#_Toc310978537)

[14. Level Design 32](#_Toc310978538)

[15. Mechanics Analysis 32](#_Toc310978539)

[Mechanics 32](#_Toc310978540)

[Entropy 33](#_Toc310978541)

[Economy 34](#_Toc310978542)

[Reward Cycles 39](#_Toc310978543)

[16. Schedule 39](#_Toc310978544)

[Developer Tasks 40](#_Toc310978545)

[17. Budget 41](#_Toc310978546)

[18. Change Log 41](#_Toc310978547)

# Executive Summary

The basic setting for Cannon in C-Sharp is essentially Industrial Revolution England with a bit of a twist. Within this world all battles are fought from castles using Cannons and other ranged weaponry. The world is also somewhat silly, being less like a realistic society, and more like a parodied version of England at that time, with emphasis on pomp and tea. The inhabitants of this world are essentially characters from an Oscar Wilde play,  living in a Saturday morning cartoon version of London.

What makes Cannon in C# truly interesting is the simple yet fun gameplay. The gameplay in Cannon in C# centres around three phases of gameplay, being fortifying, arming your fortress and attacking. It's very quick to pick up and play, requires very little time to complete a match, and due to the head to head nature, keeps players coming back for more to keep attempting to one up their opponent. Rather than be overly complicated, we opted to have simplistic streamlined gameplay with similarities to other classic games such as Rampart and Tetris to foster a sense of nostalgia in players to try to evoke memories of their childhood. Though gameplay is simple, it also takes time to master. Someone who has been playing Cannon for a while will better understand the mechanics of the game and will gain greater insight into the games strategy and mechanics, keeping them coming back for more. In doing so players will be rewarded with the feeling growth and experience the more they play the game, adding infinite replay value.

Ultimately Cannon in C# is a simple game with a lot of quirkiness and personality. It’s meant to be fun rather than complicated, and funny rather than serious and bland. Players will appreciate the comic spin that is somewhat lacking in contemporary games, bringing them back to the games of their youth. The use of head to head multiplayer is also a major throwback in these times of network play. Letting players play against each other in the same room where they can witness another players horror at their own demise is a social experience that we’re hoping to bring back into gaming. We’re not trying to create a game as much as we’re trying to create an experience for players of a new gamer generation.

# Overview

In Cannon in C# players go head to head in an intense game of strategy and warfare. A strictly multiplayer game, Cannon in C# forces two friends or enemies to compete for ultimate Cannon superiority by building fortresses and arming themselves to the teeth. Of course what would a war game be without some destruction. As such, players are also tasked with trying to destroy the opposing base faster than their opponent can rebuild. Excelling in these tasks will earn players more money to rebuild and expand towards victory, faltering will lead to destruction and defeat.

Set in an Industrial Revolution Lodon-esque society, Cannon in C#’s gameplay centres around three main gameplay aspects, those of fortifying, Cannons(arming), and attacking. To succeed, a player must be proficient in all three of these phases, or at the very least better than his or her opponent.

The first major phase in gameplay is the fortifying phase. This phase can be somewhat compared to Tetris with wall pieces. At the start of a game, each player is given half of the map which belongs to him or her. During the fortify phase, players are given random configurations of wall pieces ranging from straight pieces to corner pieces and everything in between. Using these pieces players must box in areas with unbroken walls to create fortresses. Though this sounds easy enough, the build phase is not without its challenges, as players must complete these walls before a timer expires. Though a player would naturally opt to produce the biggest fortress possible, he must balance this desire with the limited time he is given.

Now you might be wondering what one does with the fortress or fortresses created during the first phase. Well to answer your question is that it must be armed, bringing us to the second phase of "Cannons!!". Depending on a player's success during the build phase, they are rewarded with money with which to arm their fortresses. Using this money, players can buy and place Cannons anywhere within the completed fortresses they have created. Therefore the more space you have, the more Cannons you can place and the more firepower will be at your disposal. Players also have the opportunity to spend some of their hard earned cash on upgrades for the Cannons they have built. Players can choose to upgrade either the firepower, the accuracy of fired cannonballs, and the chance of earning bonuses in cash rewards. Players will also have the opportunity to build support structures which will play an important part in the next phase. These include defensive towers which shield nearby cannons and walls from the enemy onslaught, as well as healing towers, which heal nearby towers over time. In this way the game captures the essence of a tower defense type game, though with no actual enemies rushing the fortress to attack.

Finally we come to the final phase of play, and arguably the most fun of the three: attacking. Everything the player has done up until this point will either benefit them or hinder them in this phase. Here the player’s cursor transforms into an aiming reticule, and they are given only one task, aim and fire. From here players can choose to destroy their opponents Cannons, limiting firepower, or destroy walls, causing the opponents fortress to become incomplete and disabling the contained Cannons. Though largely button mashing in terms of firing the cannons, which fire one after another in succession, players must strategize and choose their targets wisely. Adding an extra element of difficulty, Cannons have a cool down, so a player who has not built enough Cannons in the previous phase may find themselves in a bit of a sticky situation come time to wage war. Though all three stages bear a similarity to the game Rampart, the attack phase is probably the most similar in that the way in which cannons fire and operate are very similar to this classic game.

Once this final phase is complete, the three phases loop again allowing for each player to rebuild and attack anew. Players continue this three stage dance of death for a set amount of turns until the money each player has accumulated is calculated and a winner is chosen, ending a gameplay session.

Now this in itself is enough to make a game, but for an added bit of flavour, the game also carries a somewhat comedic tone. Drawing upon the successes of games such as Worms and the Monkey Island series, certain cartoonish and comedic aspects are present to augment the gameplay experience. From the use of cute cartoonish graphics to represent the playing field and structures, to the use of funny voiceovers to add that extra special touch of comic mayhem, the game is much more than a simple arcade game.

Overall playing like a cross between your run of the mill tower defence and the classic game Rampart, Cannon will instil within players with a sense of nostalgia, and familiarity which will enhance their experience with our game. The twists and additions to gameplay will also serve to give players something new and increase the difficulty of the game somewhat, making it a game that’s simple to play, yet difficult to master in its fullest.

# Related Games

## Rampart

Atari Games  
 1990  
  
  Rampart was first released as a puzzle and shoot 'em up hybrid arcade game which would later influence the creation of the tower defence genre. In Rampart, players build a fortification, place cannons and shoot at enemy castles or ships. The main focus of the game is to survive waves of attacks by rebuilding and rearming a castle.

Rampart is the most influential game in the design of Cannon in C# for a variety of reasons. This precursor to the tower defence genre has been ported countless times and its simple gameplay is still appealing today. As such, Cannon in C# was born from the idea of revamping this style of gameplay with updated graphics and more in depth  tower defence elements which have become widely popular in the recent years. Thus, Cannon in C# will borrow many aspects of Rampart, namely the multiplayer gameplay, variations on its phase based gameplay, tiling system, weapons, upgrade system and voiceovers.

First of all, it is important to note that Rampart includes both single player and multiplayer modes, while Cannon in C# only inspires itself from the multiplayer mode. We found that the fondest memories of Rampart often revolve around the multiplayer mode, and that this type of gameplay would promote replayability.

Secondly, Rampart is a phased-based game with 3 distinct phases : "Prepare for Battle", "Build and Repair" and "Place Cannons". Although Cannon in C# inherits its name from the 3rd phase of Rampart, all 3 phases will be present in the game. Despite the fact that they have been renamed and the internal mechanics of each phase is slightly different, we felt that the phase-based gameplay keeps concerns separated and helps guide the player to the appropriate actions, giving them a sense of comfort and control. This is especially important when considering our casual target audience.

The first phase in Rampart lets players create their castle using tiles which are reminiscent of Tetris pieces. The tiles can be rotated and placed to enclose an era of land. In Cannon in C#, the mechanics will be very similar, but rotating the tiles is not implemented in order to provide a slight challenge in the creation of the castle.

Another feature which is obviously preserved in Cannon in C# is... cannons! The 2nd phase of Cannon in C# is focused on cannons, which are free to place on the battlefield and not awarded based on score like in Rampart. However, Cannon in C# has other types of units available to players. Tower Defence elements are incorporated through the addition of towers which fall under the defensive, offensive or support categories. This will add a level of strategy to the simple gameplay featured in Rampart. The number of different types of towers will be minimal in order to avoid alienating the more casual players.

An interesting addition in some ports of Rampart, namely the PC and SNES version, is the introduction of two “new” units. Rampart awards a certain number of cannons to be placed on the battlefield at each turn which could be traded into one of two special units, the balloon and the super cannons. In Cannon in C#, we wish to enhance this mechanic to allow players to upgrade their cannons with the score they earn through the building phase and the destruction phases. This type of upgrading scheme is very common in tower defence games, and thus will easily be understood even by casual gamers. Popular examples of this mechanic include Final Fantasy Crystal Defenders and even Plants vs. Zombies.

Finally, Rampart incorporates some voiceovers which indicate the beginning of each phase. If these were comical by design or if they have aged to be seen that way is debatable, but their effect is undeniable. Cannon in C# wishes to focus on this comical aspect and include voiceovers for phase names, but also during gameplay when selecting cannons or towers for example.

In conclusion, Cannon in C# aspires to bring a modern twist on an old classic by adding a comical tone to a simple and straightforward gameplay style. With the addition of tower defence elements to competitive multiplayer play, we want to bring a unique taste of the origins of the genre to a new generation of gamers.

## NetStorm

Titanic Entertainment   
 Published by Activition

1997

In NetStorm, a real time strategy game for PC, players direct the inhabitants of Nimbus, a world of islands in the sky, to battle each other in an attempt to capture their opponent’s high priest and acquire their knowledge. Players build bridges between islands, place buildings, weapons and send troops to gather resources in this game with complex tower defence elements.

Although NetStorm is  far more complex than Cannon in C#, the two titles will share some similar elements such as multiplayer gameplay, strategic use of tiling systems, offensive, defensive and support gameplay elements, upgrading systems as well as in game voiceovers.

First of all, NetStorm owed its popularity to its multiplayer mode which enabled simultaneous play with up to 8 players. We feel that multiplayer games are still relevant today in our world where social interactions happen more and more through the Internet. However, the multiplayer in Cannon in C# will be handled differently from NetStorm. Cannon in C# will support 2 player local multiplayer only, which is something that has been significantly left out of most modern games while there is still demand for such a feature.

Subsequently, the tiling system in Cannon in C# and NetStorm have much in common. Cannon in C# uses tiles of various shapes to create fortifications, while NetStorm uses the tiles to create bridges between islands. The tiles in both titles are “free”, such that players can use as many as they desire without restriction of time or cost. The similarities do not stop there, as the placement of the tile is highly strategic in both games. The area enclosed by the tiles in Cannon in C# defines the area in which cannons can be placed. In Netstorm, the shape of the bridges created define where various gameplay elements can be positioned. We feel this strategy for altering the gameplay area creates a slight challenge while staying simple enough not to discourage casual gamers. The tiling system also creates a sense of territory for players to defend in both games, which is characterized by the temple in NetStorm and the castle enclosure in Cannon in C#. Indeed, players can only build units on an island if they have built a temple on it, and similarly, in Cannon in C#, players can only build a unit within an area enclosed by walls. NetStorm’s tiling system was a key feature which helped set it apart from its competitors, and we feel that a similar “Tetris shaped” tile system will help Cannon in C# set itself apart from other tower defence games.

Furthermore, NetStorm boasts an incredible number of building, troop and weapon types. Although Cannon in C# does not aspire to include as many different kinds of units, it does hope to bring some strategic aspect to a simple gameplay style reminiscent of games like Rampart. While NetStorm uses the 3 furies (Wind, Rain and Thunder) as elemental forces which power units and implements various roles for the units within a same elemental alignment, Cannon in C# will focus on the simple defensive, offensive and support roles of different units. Although the interaction between buildings and units in NetStorm is very interesting, we consider that it is too complex and time consuming to understand and master for casual players to enjoy.

The upgrade system is also a feature present in both titles. NetStorm allows players to upgrade their workshops using the resources they collect from geysers, while Cannon in C# will allow upgrades to be purchased using in-game currency. This currency will be acquired at the end of each round as players destroy enemy walls, cannons and towers. Although the currency used is different, both upgrade systems are very similar: they allow players to upgrade existing units to increase their effectiveness. A neat feature that Cannon in C# might want to borrow from NetStorm is the delay created when upgrading units. For example, upon upgrading a workshop, it is unusable for a short period of time and cannot be selected for further upgrading until the first upgrade has been completed. This mechanism could enable the game to refrain players from upgrading their cannons or towers repeatedly within a short amount of time and gaining too much advantage over their opponent.

Finally, NetStorm includes voiceovers for in-game troops. Upon selecting an action, the unit will respond to the player with a “Yes Master” or similar statement. This mechanic is similar to what we wish to implement in Cannon in C# where upon upgrading or placing cannons, certain voiceovers or comical comments will be heard. With the tone of NetStorm and Cannon in C# being drastically different, the style of voiceovers will not be similar but the concept and timing for the sounds will be.

In conclusion, NetStorm is an inspiration for multiple features in Cannon in C#. Although the complexity of NetStorm is not something that Cannon in C# will replicate due to the difference in target audience between the two, many gameplay mechanics apply to both titles. The always popular multiplayer gameplay mode, the unique tiling system, upgradable tower defence units and unique voiceovers will make Cannon in C# attractive and unique while using proven tower defence mechanics to ensure an intuitive and playable game.

## Worms

Team 17   
 1995  
  
 Worms is a turn based strategy game wherein players control teams of worms, to do battle against each other using a massive arsenal of quirky weapons and abilities. Players attack each other, trying to eliminate the opponent's worms first. This is achieved through either reducing each worms health points to zero, or through knocking the opposing player's worms off the stage.  
  
 In terms of gameplay, Worms and Cannon in C# are not very similar. Though the basic premise of opponent elimination is there, what is really borrowed from Worms is the game's art style, voiceovers and general quirkiness.  
  
 As previously stated, the main goal of worms is enemy domination and elimination. Though this is also true of Cannon in C#, it is still quite different. Whereas in Worms you achieve this goal simply by attacking your opponent mercilessly until you or he are eliminated, Cannon in C# requires players not only to destroy, but to build and improve. As such, victory in Cannon in C# is not primarily through elimination, but through a points system. Though destruction is still a big part of the game, it is not the only key to victory.  
  
 A large part of what made Worms an interesting and fun game, was its aesthetic. The game looked like something pulled from an eighties Saturday morning cartoon. With vibrant colors, silly character animations, and comical weapon designs, it was truly a unique gameplay experience. The overall aesthetic of Cannon in C# is very similar to Worms in this way. The colors and overall look of the game was inspired by the art style of Worms, from the map itself down to the look of the Cannons. Because of its style, Worms sets a comedic tone, which has endeared it to gamers of all persuasions, from casual to hardcore. The art style of Cannon in C# is designed for the same purpose.  
  
 Another artistic element that made worms so unique was the use of humorous voiceovers and sound effects. In Worms, most actions triggered a vocal reaction from other worms. These sound bytes were comical in tone, and added personality to the otherwise unremarkable worms. The game also had a set of voices which players could choose from, ranging from angry Scotsmen, to evangelical preachers. Not only did this add customizability to the game, but it also further cemented its tone as comical and quirky. In Cannon in C#, we use sound bytes in the same way. In using comical voices and sounds during gameplay, we hope to capture the same quirky tone as Worms.   
  
 In conclusion, though Worms will have no real heavy influence on Cannon in C# in terms of gameplay or mechanics, it is still the driving influence behind the tone of the game. The art style, and voiceovers in Worms will be a huge inspiration for the quirky style of our game, and add much in the way of aesthetics.

## Tetris

Alexey Pajitnov

1984  
  
 In Tetris, a world renowned puzzle game available on many different platforms, players are charged with arranging vertically falling blocks of various shapes into horizontal lines. Upon the completion of a line or multiple lines, the completed line(s) disappear and the player is awarded points based on the number of lines completed.  
  
 Though the gameplay of Tetris and our envisioned game are not similar, we are drawing on the block system and line creation for the building element of Cannon in C#.   
  
 Creating unbroken lines in Tetris was paramount to success in the game, as well as being the only real objective. In Cannon in C#, we would like to apply this principle to the building phase. Rather than creating unbroken horizontal lines, it would involve creating unbroken lines in the form of walls. By enclosing an area by unbroken walls, you create the area in which players would be able to build cannons. Though the use is not very similar, it does draw elements from this gameplay aspect.  
  
 The block system of Tetris on the other hand is more of an aesthetic throw back. In using similar blocks as those used in Tetris we hope to evoke a sense of nostalgia in players, as well as drawing upon the players familiarity with Tetris and its line system.  
  
 In conclusion, though the similarities between Tetris and the game we hope to create are few, by drawing on certain key gameplay and aesthetic elements we hope to use players memories of this classic game to instil a sense of familiarity within them.

## Monkey Island

Lucasarts

1990  
  
 In Monkey Island, a point and click adventure game for pc, players were tasked with controlling wannabe pirate Guybrush Threepwood in his quest to defeat the evil undead pirate leChuck. As in other point-and-clicks, the gameplay of Monkey Island is centered around moving Guybrush from location to location using mouse clicks, and manipulating objects and the environment. Monkey Island is also very well known for its comedic tone and art-style.   
  
 Since we do not intend to make a point-and-click game, absolutely no gameplay mechanics in Cannon in C# are influenced by Monkey Island. Rather what we are drawing from this classic game is the tone and to a lesser degree, the art style.  
  
 The humour and tone of the first Monkey Island game were some of its defining features which endeared it to a generation of gamers. As such, it is a major source of inspiration for our idea. Though our game won’t have dialog, we hope to set a similar tone through a humorous backstory, and funny voiceovers.   
  
 The art style of the Monkey Island games is also of note. Being colourful and cartoonish, it further helped cement the light hearted tone of the game. Our vision for Cannon in C# is likewise geared towards cartoony colourful artwork which would set it apart from current trend of more realistic games with a penchant for earth tones.  
  
 In conclusion, Monkey Island and its sequels are a great inspiration for Cannon in C#, though solely in terms of tone and aesthetics. Due to the rather specific mechanics of point-and-click adventures, no gameplay elements could be used as inspiration for the mechanics of Cannon in C#. Rather we hope the use humour and of a light-hearted tone will endear players to our game, like it did for Monkey Island all those years ago.

# Player Composite

## Casual Gamer Profile

Erica Brydon is a 22 year old university student who lives with her parents and has a part time job as a clerk in a chic fashion boutique downtown. She loves fashion, going out with her friends, shopping, TV shows like Grey’s Anatomy and her iPhone. She plays games  like Angry Birds and Diner Dash on her iPhone while she is out and about.  She also plays a few games like FarmVille and Diamond Dash on Facebook, a website some would say she is addicted to. Family is very important to her, and she likes to dedicate time at least once every 2 weeks to playing with her little brother for a few hours. Being a young boy, he likes to play his family friendly games on the game systems on the family room TV. Erica joins him in simple multiplayer games and although she would never admit it, she enjoys playing games from the LEGO and Rayman Raving Rabbids series with her little brother. She enjoys the humour and the cuteness factor in those games.

## Transitional Gamer Profile

Andrew Barnes is an 18 year old student who is on his school’s football team. Popular with the ladies, Andrew likes to party and spend time with friends. In the recent years, he has been introduced to games like Rock Band, Dance Central and DJ Hero at friends' house parties. More recently, Andrew has been playing some first person shooters at a friend’s house and has been enjoying it enough to consider buying an Xbox 360. Andrew does not spend much time on his studies and gets bored easily of activities. His short attention span makes him favour short first person shooter matches, high intensity sports and cheaper games and activities. Andrew enjoys playing games to distract himself, to relax and have a good time with friends. Andrew likes violent, gory or war related games and sports related brands and merchandise.

## Hardcore Gamer Profile

Tamas Langsner is a 30 year old programmer, who is single and lives on his own. He does martial arts as a hobby and is passionate about video games. He owns every game system of the current generation, and cannot bear the idea of parting with his dear Nintendo and Super Nintendo. Although he is primarily a fan of shooters  and owns all the latest games, he still has a soft spot for arcade style games which he played growing up. Tamas lives a simple life, and prefers to spend his hard earned money on new technology and games. He plays games almost every night of the week, often online with his friends. He owns a powerful gaming PC, but still prefers playing his Xbox 360 on his HDTV. Tamas always loved playing games and still loves them today, both because of nostalgia and because of the escape they provide. For him, games are a social activity which enables him to keep in touch with his friends as well as a way for him to wind down on his own after a long day at work.

# World

## Setting

Era : Primarily 1830s England, but also all over the world.    
Theme: Quiet and not-so-quiet English countryside.   
Tone: Comical, slapstick   
  
England: 3 o’clock tea!  
Beijing : 11 pm  
Delhi : 8h30 pm  
Huston: 10h00 am

## Backstory

Once upon a time, in the quiet English countryside, Sir Charles C. Sparkington was sipping on tea in his study on a beautiful rainy afternoon. Weary after a long day of research, Charles, who’s wife would surely point out that he always kept his study immaculate, did not bother putting away his research material before enjoying his delicious cup of Earl Grey tea. In a fateful instant, lightning struck outside the opened window of his study, causing Sir Sparkington to spill the perfectly brewed beverage all over his desk and his pantaloons. An ink bottle rolled, a cucumber sandwich was submerged, gunpowder flew into the air, Petri dishes flipped and an entire day’s worth of research was tossed together with tea as dressing. Even Sir Sparkington himself could not tell you exactly what went into that mixture; all he knew was that something incredible came out of it. Before he even had a chance to get a change of pantaloons, the mixture had a life of its own. As Sir Sparkington looked on in terror, a strange liquid seeped its way out of the chaos of his desk and leaked onto the floor of his study. It wiggled and squirmed, twisting and twitching with newly created life.

Meanwhile, in a small store in Beijing, an old man spilled his tea onto his late night snack and the herbal medication he was preparing for a customer. At the exact same time, a woman in Huston spilled her breakfast tea over the laundry containing her husband’s coal covered work clothes. Simultaneously, in one of Delhi’s bazaars, a little boy spilled his father’s spice tea over all manners of merchandise in the back of their small family owned stall.

All over the world, tea was spilled right as Sir Charles C. Sparkington ruined his pantaloons. And all over the world, the same strange phenomenon occurred....

A new life form took shape and it looked just like a ….  Cannon?!?

As news of this surprising discovery spread from all corners of the world, disagreement arose over whose discovery it was.

Noticing that the new cannons could indeed shoot cannonballs, Sir Sparkington had the brilliant idea of summoning his fellow inventors to an “old school” battle to settle the score for the rights to the invention.

Thus, people from all over the world gathered at Sir Sparkington’s estate, in the no longer quiet English countryside, to battle with Cannons... for Cannons!

### Premise for gameplay

Citizens from nations all over the globe come to take part in an “old school” battle with their Cannons. They must build fortifications on Sir Sparkington’s estate to protect their dear invention. They then proceed to place as many cannons as they can to blow their enemies away!

Gameplay can eventually take players all over the world, where more duels can be held, giving the game a comical worldly feel.

# Characters

### Sir C. Sparkington

Fun Fact: The C. stands for Cannon!!  
  
 Sir Sparkington, the rumoured inventor of these new-fangled Cannons, is somewhat of an interesting character. Born to a lovely doting mother and strict General father( who smelled suspiciously of elderberries I might add), Sparkington was of a slightly odd pedigree. Though his mother was quite mundane, spending her days drinking tea with other equally boring companions, his father was of a most respectable sort. A military man by trade, he insisted that the young Sparkington enlist as soon as possible. After getting him a commission in utero, the elder Sparktington was quite confident that his young son would develop into a fine young soldiering lad like he had. Alas it was not to be.   
  
 Sparkington the younger grew to be nothing like his father. Rather foppish by nature, he had no more a militaristic inclination than a soiled handkerchief. Spending his days sipping tea along with other dandies, and nights carousing with women of the most scandalous persuasion, he became a perpetual disappointment to his upstanding father.   
  
 The relationship between the two men became more strained as the years went on until Sparkington the younger was caught up in a terrible scandal ( Though it is too terrible to recount here, let me just say it involved a cucumber sandwich, a buttonhole, and a private room at the Savoy). This final shamming was too much for the elder to bear. Realizing something must be done about his wayward youth, he decided that a proper education might just be the key.   
  
 It was not long before Sparkington the younger found himself a student at the Inventor’s Academy of Chemistry and Kerfuffles, studying the sciences of machinations and chemical reactions. It was here that he discovered his love for science (and cake) and decided to pursue this noble career wholeheartedly. Upon graduating he took a wife and began a long life of toiling away blissfully until...

### Cannons

Cannons form a strange new race of living organism that was born from a mix of crazy circumstances and human clumsiness. Each Cannon is unique and has its own personality. They come in a wide range of patterns and colors so everyone can find a Cannon to fit their taste. Although they cannot communicate with humans, they have attempted to do so in many occasion through cannonballs and explosion sounds. Although Cannons are still mysterious and misunderstood, one thing is clear: Cannons just want to be loved!

# Progression Graph

The following diagram represents the flow of Cannon in C#. From the main menu, the player can select to start a game, view the options or exit the game. Once a game is started, players will select the length of the battle and the color of their Cannons, amongst other things. Once selection is confirmed, the players will start the Fortify phase, which is followed by the Cannon phase and the Attack phase. Gameplay can be paused at any point during these phases and the players can be returned to the main menu. Once the 3 phases are complete, the players will view a scoring screen and the 3 phase process will start over until the predefined number of iterations has been reached. Once the number of rounds to complete is reached, the players will be shown the final screen where a player is crowned winner. From this screen, the players can return to the main screen.

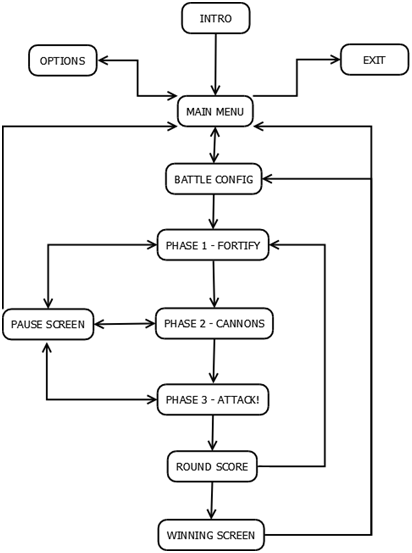


Figure : Cannon in C# Progression Graph

# Art Direction

## Visual Language

### Shape and Silhouette

During the design process, great care was put into the choice of silhouette for all gameplay elements. The art design team realized early on that each Cannon, tower and wall piece should be instantly recognizable by its silhouette to enable players to quickly grasp the differences between elements and enable quick and engaging gameplay.

A smooth organic shape was chosen for Cannon silhouettes in order to reflect their backstory. Furthermore, the rounded edges and smooth surfaces of the organic Cannons give them a “cute” look to make them endearing to casual gamers.


Figure : The silhouettes of the 3 levels of Cannon upgrades

The towers and walls, which are not considered organic,  were given for the most part very angular shapes and silhouettes in order to contrast with the organic shape of the Cannons. As can be seen below, the tower design is still in progress, as the attack buff and the healing tower both  have the same silhouette. Further work is required for both towers to have differentiable silhouettes.

###### https://lh4.googleusercontent.com/-WWLUfeFx04nftFjaDh4EGFZM-89y4A6sPZFgjRlILzoByEcTrhIpTM0vBSxOKOS3S2voDnAWoVL7GAtjsN35LPvrTHsxsiw_VC4Cog_COxAug2uHqshttps://lh6.googleusercontent.com/9hQCqMRMS52PValMMdzdR6SQO46Q3esUEri30x1Ud8clKXgmSyY5P4auWpLMTSmNpVnbdwx5JBGsv9TiOQIjin31NJaBHq7WK-OLQL66dC0WhRHvggw

Figure : Silhouettes of the towers

### Proportions

Proportions were taken into account once the silhouettes for gameplay elements were selected. To put emphasis on the Cannons which are to be the focus of the game, the Cannon sprites will be oversized when compared to the rest of the world. The increased size of the Cannons also give an impression of weight and robustness.





Figure : Screenshot showing palceholder art and proportion between different gameplay elements

Characters, including inventors and Cannons, have disproportionately large eyes and heads, again to increase the “cute” factor and make them more attractive to a wider audience. The characters and Cannons in the eventual intro for the game should be larger than the environmental elements in order to keep the focus on them, and to give the intro a picture book feel.

### Orientation

The point of view and orientation of characters and gameplay elements in Cannon in C# is fairly simple. During gameplay, the view is always top down while the introduction is essentially a side view reminiscent of children’s picture books. As can been seen in previous screenshots, the top down view enables players to share the screen space and see all units appropriately. The side view for the picture book introduction is the best suited view for the whimsical style of the backstory.

### Color Palette

The chosen colour palette reflects the cartoony and comical tone of the game. Here is a brief description of the choices pertaining to the select hues, values and saturation levels for each element.   
  
 First of all, special attention was given to the value of each element when preparing the sprites. The game was adjusted so that a grayscale version of the title could would still be playable. This way we ensure that players who are color blind can still enjoy the game without undue confusion. The Cannons and towers are mostly of higher value than the walls and thus are more easily distinguishable on the battlefield. Some of the faction’s Cannons have very similar values and may be hard to differentiate, but we feel this is a non-issue as the field is clearly separated and the enemy Cannons will never appear close to each other.

Secondly, the main scheme for the hue selection was complementary. Each faction uses one main color for its Cannons and its complementary hue for the details on the sprite. The towers also use both a subset of the faction colour palette as well as a subset of the environment colour palette (ie: walls). This enables them to be recognizable as belonging to one faction while blending in nicely with the walls of the fortifications. The overall colour scheme feels warm based on the hues decided upon for the dirt, walls and grass; i.e. their hues tend to lean towards “warmer” colours with a red/yellow tint.

Finally, the saturation of the Cannon is higher than that of the walls and towers, which makes them contrast with the dull grass, dirt and tiles allowing them to “pop out” and making them easier to find on the battlefield.

### Motion

Motion is very limited in Cannon in C#. During gameplay, the Cannons rotate on their bases in order to aim at different areas of the enemy's fortification. The cannonballs also move from one side of the battlefield to the other in a straight line. As for the introduction, the characters are still, like images in the picture book. Thus animation is limited and characters do not require special moves.

### Depth

The depth cues used in Cannons in C# are very straightforward. Given that there is only one camera angle during gameplay, the main depth cue used during is the desaturation of the background (ie: grass tiles) and the addition of a thin black outline around each unit on the battlefield.

## Reference Art

Limited reference art was used for Cannon in C#. The main inspirations were the original Rampart game for the setting as well as Worms for the colors, shapes and sounds. All references were made directly in game and no reference art still images were kept.

## Concept Art

### Cannons


Figure : The evolution of the Cannon concept art, from the original design to the final product, through intermediary shadow/shape studies.

### Towers



Figure : Tower Concept Art

### Font & Text

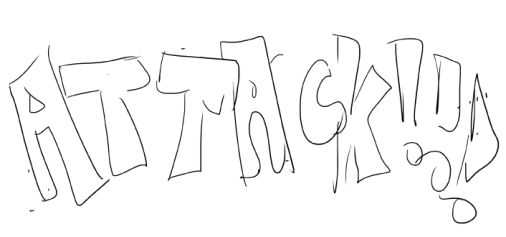


Figure : Evolution of in game and main menu font

# User Interface Storyboards



Figure : Main Menu

This screen shows an early storyboard drawing of the title screen of cannons. It sports a very simple interface with several options and rolling hills in the background.

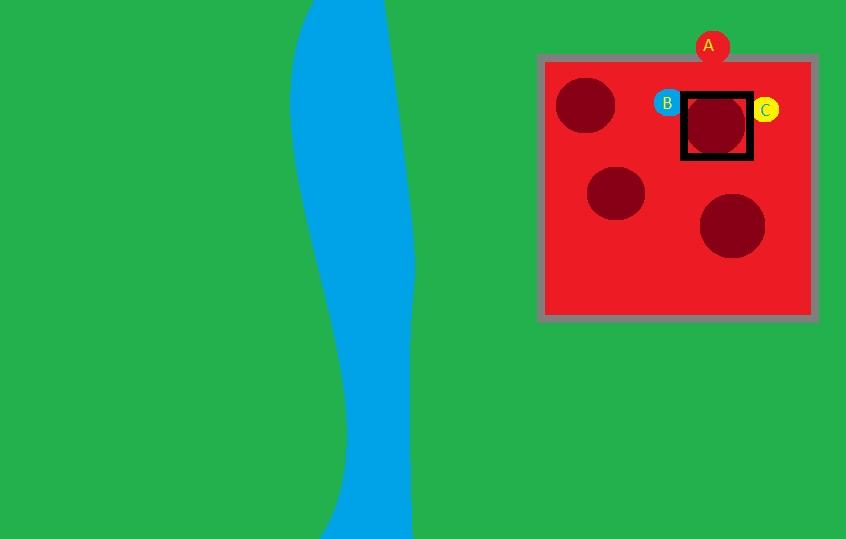


Figure : The Cannon Phase. The cursor selects the area to place a Cannon, or if over a Cannon, offers the possibility to upgrade it using the displayed buttons

This screen shows a storyboard drawing of the placement and upgrading of cannons. Of note are the square cursor around an already placed cannon and the pop up showing which buttons to press to upgrade the cannon in a desired way.

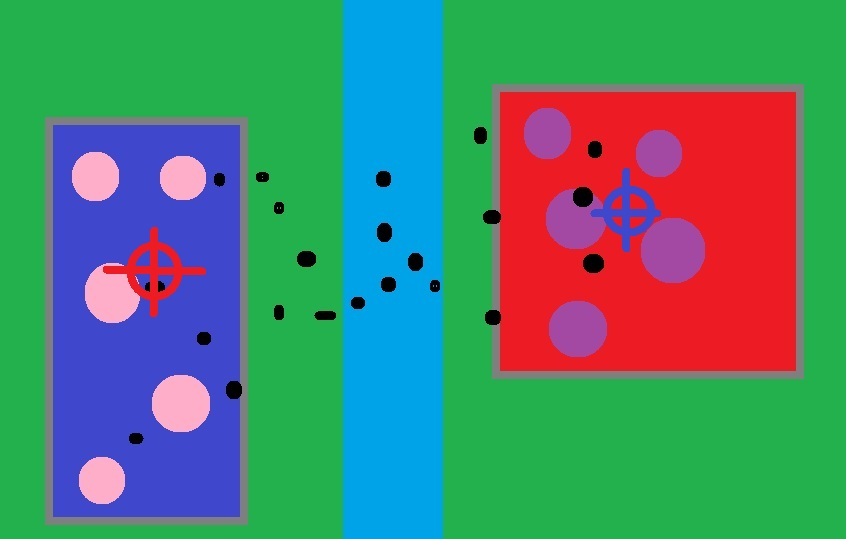


Figure : Attack Phase

This screen shows a very early storyboard drawing of the attack phase. Note the colored cursors pointing to the opposing factions fortress and the cannonballs in midflight.

# Tags and Dialogue

The following is a list of tags used in the source code to identify various assets such as sound and text. The tag name, its description and the current asset associated with the current development build can be found in the following table :

Sound Tags

|  |  |
| --- | --- |
| **Tags** | **Description** |
| *Example* | *Description of the tag* *(Current asset associated with tag)* |
| SplashBGM | Background music for Logo Splash Screen (Pachelbel Cannon in D) |
| TitleBGM | Background music for Main Menu Screen (Pachelbel Cannon in D Rock version) |
| Phase1\_PlaceWall | Sound for placing walls (Hammering on wood sounds x 6) |
| Phase1\_EnclosedArea | Sound for Enclosing an area (Jackhammer sound) |
| GameplayBGM | Background music for gameplay (Warcraft 2 : Alliance bgm 1) |
| MenuSelected | Sound played when menu item is selected (Cannon fire sound + man screaming “FIRE!” or “CHARGE!”) |
| MenuSelection | Sound played when changing selection in a menu (FF7 menu selection sound) |
| Cancel | Sound played when cancelling a menu (FF7 cancel sound) |
| CannonFire | Sound when a cannon is fired (Cannon firing sound) |
| BuildingDestruction | Sound when a Cannon or Tower is destroyed (Building collapse sound x3) |
| SimpleDestruction | Sound when a wall is destroyed (Small explosion sound) |
| CannonHit | Sound when cannot hits a structure (Heavy object hit sound) |
| DamUpgrade | Sound when upgrading Damagitude (Warcraft 2 - Human Foundry and Alchemy sound) |
| AccUpgrade | Sound when upgrading Accuracy (Warcraft 2 - Mine and Human blacksmith sound) |
| ChnUpgrade | Sound when upgrading Chanciness (Warcraft 2 - Construction and Human Lumbermill) |
| WormTaunt | 10% Chance of sound playing while player 2 firing (Worms taunt - Incoming, Laughs, Surprise, Watch This, You’ll Regret that, Special Delivery, Fire, Ill get you) |
| WarTaunt | 10% Chance of sound playing while player 1 firing (Warcraft 2 Dwarf sounds - Ill blow things up, Bombs are great, Got one back with me dog, Welcome to my nightmare, I am Iron man, Time to die, Do you feel lucky Punk) |
| Chanciness | Sound played when player receives gold bonus in attack phase (Mario coin sound) |
| Coins | Sound played when player receives gold during the end of the round (Sound of coins dropping on table) |
| READY\_INSTRUCT | Text indicating what button to press when player is ready for next phase. (Press RB when READY!) |
| VERSION | Version of the game (v0.8) |
| BUILD\_TYPE | The type of build of the current release (ALPHA RELEASE BUILD) |

# Technology Plan

The technology used for this project can be divided into the following categories: Game and Gameplay Code, Development Hardware,  Artistic Design, Documentation, and Management/Asset Management.

For game and gameplay source code, the team will use Microsoft Visual Studio 2010 and Microsoft XNA framework 4.0. The only code reused in the project is the Gamestate Management code sample from App Hub used to manage the main UI screens at the beginning of Cannon in C#.  Original code can be found at:

http://create.msdn.com/en-US/education/catalog/sample/game\_state\_management.

The hardware used for development includes a few laptops and personal computers with Visual Studio 2010 and an Xbox 360 for deployment through the XNA Creator’s Club and XNA Game Studio.

Concept art will be done using Adobe Photoshop CS4 and art assets will be completed in Adobe Illustrator CS4.

Documentation will be created in Google Documents and formatted and published using Microsoft Word. Microsoft word will also be used to keep the minutes of the development team meetings.

Finally, assets will be managed using Tortoise SVN combined with a free XP-Dev subversion  repository (www.xp-dev.com). Art assets will be shared through Dropbox (www.dropbox.com).

# Software Architecture

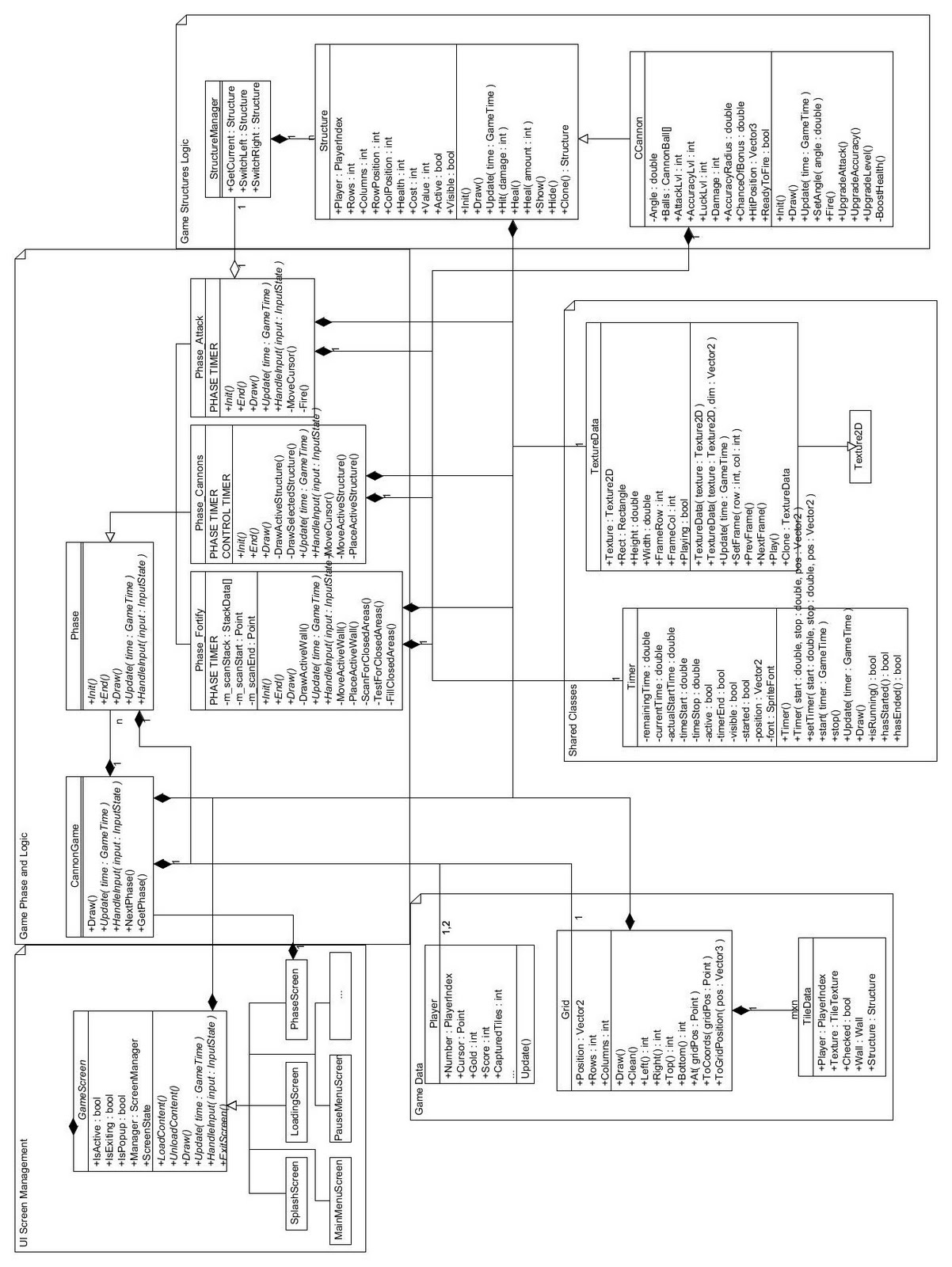
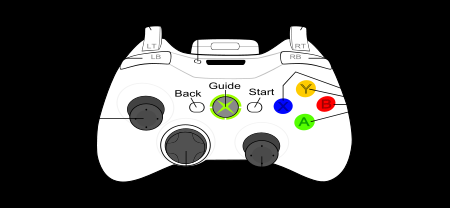


Figure : Class diagram for Software Architecture of Cannon in C#

The CannonGame class essentially represents one running game, with a chosen amount of phases, and is created when the players choose to play from the main menu. It maintains a Phase object, which is initialized to the proper Phase subclass. The UI passes players to the CannonGame, and it subsequently creates a Grid, and passes the grid and those players to each new Phase. Each Phase takes care of its own game logic by using the Player and Grid objects, which are mostly just data containers. This is to ensure that if ever there is a future desire for online multiplayer compatibility, the data which must be persisted are in the smallest objects possible.  
      
  All textures are stored in TextureData objects, which potentially store them as tilesheets of fixed rows/columns and dimensions. The class also supports playing animations. A ContentManager loads, unloads, and makes TextureData objects accessible using the string name of the texture.  
  
  Finally, the StructureManager allows for scrolling through various types of Structures. Although, at the moment, the game only allows for Cannons to be created, the system has full support for the addition of towers. Input in the Cannons phase is also hooked up (though disabled) to scroll through each purchasable Structure.

# Controls



Main/Pause Menu  
  
  
Directional Pad -     Move selection  
Left Analog -     Move Selection  
A Button -     Select  
Start -        Select  
Back -        Cancel/Go back

Fortify Phase  
  
Directional Pad -     Move wall piece  
Left Analog -     Move wall piece  
A Button -     Place wall piece  
Start -        Pause game  
Right Bumper -         Player Ready  
Left Bumper -         Skip Phase (when Debug mode active)

Cannons Phase  
  
Directional Pad -     Move cursor  
Left Analog -     Move cursor  
A Button -     Place Cannon (when on empty space)  
B Button -    Upgrade CHA (when on placed cannon)  
X Button -    Upgrade ATK (when on placed cannon)  
Y Button -    Upgrade ACC (when on placed cannon)  
Start -        Pause game  
Right Bumper -         Player Ready  
Left Bumper -         Skip Phase (when Debug mode active)  
  
  
Attack Phase  
  
  
Directional Pad -     Move firing cursor  
Left Analog       -     Move firing cursor  
A Button      -     Fire  
Start          -    Pause game  
Left Bumper -         Skip Phase (when Debug mode active)

# Level Design

Cannon in C# only includes a single setting and there is no true level progression. The game is based on phases as described in the Overview section of this document.

Since the gameplay takes place on the same standard field every round and that the modifications to this field are entirely user generated, there is no true randomly generated level design and there is no true choke point.

Each user is assigned a side of the playing field to build on, thus a given player will always build on one side of the screen and attack the other.

# Mechanics Analysis

Mechanics  
  
    Cannon in C# combines a few distinctive mechanics. Shooting, character building/tower defence, some geometry mechanics and racing mechanics will all be used to some extent in our title. Each of these mechanics play an important role in creating balanced and engaging gameplay. This section describes the reasoning behind the use of each mechanic and briefly describes the way we balance the in game economy.

Before delving into the mechanics Cannon in C# is to implement, it is important to note that certain balancing problems don’t apply to the gameplay we want to create.

First of all, Cannon in C# is set to be a real time strategy and shoot 'em up game where players play simultaneously. Thus, there is no “first move” advantage for any player. Secondly, there is also no real way to rush in Cannon in C#, as the end game condition is not the same as the victory condition. Consequently, there is no real need to implement any rush prevention mechanisms.

Along the same lines, there is no true super unit in Cannon in C#. Although some could consider that the fully upgraded Cannon is a super unit, we feel that they are not, as they as available early enough in the course of gameplay and do not constitute a guarantee of victory. The downside to the decision to make fully upgraded Cannons available early in the game may be that the novelty of using them will wear off quickly since players will use them frequently during a single game. We feel we can counter this loss of novelty with other engaging gameplay mechanics.

The gameplay mechanics that Cannon in C# wishes to implement includes Racing. This is implemented through the constant use of a timer/countdown in each phase of the game. This gives a sense of stress and urgency which will keep the player engaged in each phase. This mechanic was used in Rampart for the same reasons. In keeping with the spirit of Rampart, we are reusing this mechanic as well.

Another mechanic which was also included in Rampart was shooting. The shooting mechanic is fairly straightforward in Cannon in C#. The players aim their cursor at an enemy unit and shoot using all available Cannons to destroy the target. The shooting mechanic is very much based on the twitch reflex skill level and will keep players with that skill engaged and entertained. The shooting mechanic is overall empowering for players and keeps them engaged using sounds and animations of explosions. Cannon in C# wants to use these sounds and animation to keep the player engaged but also as flavour elements to help bring the gameplay mechanics and the setting together.

A mechanic that will be engaging both for players with the twitch skill and those without is the geometry mechanics of tile placement and matching. The "Forfity!" phase in Cannon in C# will include fortification mechanics similar to Rampart, which require the player to position Tetris style pieces onto the battlefield to create an enclosed area for Cannon placement. This mechanic takes after Tetris in some ways, bringing the matching mechanic into play, while the fact that players shape the battlefield as they place castle walls is more reminiscent of a tile placement mechanic. Either way, we feel that success in the building phase in Rampart and in the "Fortify!" phase in Cannon in C# depend on both the twitch reflex and the strategic skill of players. This mechanic will thus help keep different kinds of gamers engaged in gameplay.

Finally, the gameplay mechanic that balances the twitch-reflex-heavy shooting mechanic  is the unit building/tower defence mechanic. The depth of this particular mechanic is what we feel makes Cannon in C# a spiritual successor to Rampart and not a Rampart clone. The possible emergent nature of this mechanic ties is well with our arcade style game. Although emergent gameplay cannot be designed easily, we feel that the addition of enough variety in the ways upgrading the Cannons and towers gives room for this behaviour to manifest itself. Players will be battling each other through multiplayer gameplay but will also discover that they are fighting the game and its economy in order to build the best Cannon configuration and gain a strategic advantage on their opponent. The discovery of new strategies with time and in game experience will create a more emergent gameplay style. Not only is the tower defence mechanic increasing the emergence of Cannon in C#, it is also going to be used to tie the gameplay with the setting and backstory. Level specific sounds can be generated and different names and appearances for upgraded Cannons will  help define the setting and the tone of the game. In Cannon in C#, the tower defence elements and combat simulation is kept fairly simple to accommodate casual players but will offer enough variety to allow for experimentation, unlike the very limited unit variations in Rampart.

### Entropy

Since the end game condition is a set number of rounds decided upon by players before engaging in gameplay, we are implementing a relentless entropy. This can cause players to play in a more strategic fashion which is intended in the case of Cannon in C#. The strategic elements are deliberately kept simple to help the casual gamers feel comfortable, but strategy is definitely an aspect of gameplay that the development team wants to encourage. Although the premise of the game and the gameplay itself may seem simplistic, adding the relentless entropy mechanic can help us make the strategic gameplay more emergent. Furthermore, the end-game condition (# of rounds completed)  is not the same as the victory condition (highest score), which helps players stay engaged throughout the entire game and even during the last round, because there is a sense that changing the outcome of the game is always possible.

Economy  
  
  Before discussing the economy of Cannon in C#, it is important to remember that the target audience is casual, and thus the team is not expecting extensive study of the economy system by players. We feel that the simple and fairly robust economy will prevent players from attempting to find exploits in the following system. Although exploits are not ruled out by the team, they should not be so severe as to break the game, and may only serve as an additional strategic aspect of Cannon in C#. We feel that restraining  and controlling the economy too much would reduce the emergent aspect of our title.   
  
  Cannon in C# includes 1 liquid commodity, the gold, and 2 types of frozen commodities, the Cannons and towers. Although this is less than the usual recommended 4 liquid commodities, we feel that such a complex economy would alienate more casual players and would distract from the very fast paced gameplay of Cannon in C#. The Cannons will have 3 different attributes that can be upgraded and the towers only 1, their level. This system seems to have enough variance, as playtests have shown that losing players can easily recover and that an advantage in one round does not necessarily mean an advantage in all subsequent rounds. Positive feedback exists in the form of a high score giving more gold, which enable players to buy more upgrades for their Cannons, making it easier for them to get a better score in the subsequent rounds. However, it is important to note that this positive feedback is balanced by the fact that Cannons can only be positioned within enclosed areas. This sometimes seemingly harsh restriction on the number of Cannons or towers prevents players from spending all their gold on upgrades and thus getting too much power.

The gold players obtain should increase exponentially as the game progresses. As players build better Cannons, they are worth more when destroyed and thus, the score and gold gains will be much more important towards end game, making it extremely engaging since the fate of the players is not sealed even in the last rounds of the game. The gold of each player will increase at the end of each round, and then decrease during the Cannon phase due to the upgrades. Overall, the amount of gold leftover after the upgrading phase should become increasingly important as the game progresses, to allow for more substantial purchases like towers and tower upgrades.   
  
 The rest of this section explains the choice of the values for the measurable properties of the game. Please note that value refers to the amount of score/gold received upon the destruction of an element, while the cost is the amount a player has to spend in gold to acquire the unit.

**Walls**  
Base HP:  40   
Base Wall Value: 50  
Cost : Free

**Cannons**  
Base Cannon HP: 50  
Base Cannon Damage: 10  
Base Cannon Value: 100  
Cost: Free

Cannon Upgrade Effect Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Cannon Level** | **Chanceiness** **Multiplier** | **total%** | **Attackitude** **Multiplier** | **total** **dmg** | **Precisionation** **Multiplier** | **pixel** **radius** |
| **Level 1** | (base - X1 Multiplier) | 5% | (base - X1 Multiplier) | 10 | (base - X1 Multiplier) | 40 |
| **Level 2** | X 1.5 | 7.5% | X 1.5 | 15 | X 1.5 | ~25 |
| **Level 3** | X 2 | 10% | X 2 | 20 | X 2 | 20 |
| **Level 4** | X 3 | 15% | X 3 | 30 | X 3 | ~14 |
| **Level 5** | X 5 | 50% | X 4 | 40 | X 5 | 8 |

The above table shows the 3 attributes that affect Cannon effectiveness on the battlefield. Note that the use of quirky names is intentional, and considered to be flavour text for the game, helping to tie in the tone and mechanics together.   
  
 The first attribute is “Chanceiness” or Chance, which is described as a percentage chance to gain a score bonus upon destroying an enemy Cannon with a chosen Cannon. In other words, a given Cannon has a chance to generate a score bonus upon destroying an enemy Cannon. Each Cannon can have a different level of “Chanceiness”.   
  
 The second attribute is the “Attackitude” or Attack Damage. This attribute is fairly self explanatory, with each upgrade increasing the damage a cannonball inflicts upon impact.

  The last attribute is “Precisionation” or Precision. This represents the precision with which the Cannon aims. It is described in a pixel radius. Although the player will always aim using the same cursor in game, the Cannons will not always hit the target at the exact location the player is aiming. This adds a level of realism and some element of chance to the gameplay. As such, the player can improve the level of precision of their Cannons which effectively reduces the radius in which the cannonball can fall. The exact location where the cannonball falls is a random function which generates a point within the defined radius.   
  
  A lot of thought has been put into the multipliers for each upgrade. From experience, we noticed that simple multipliers that are the same (or very similar) for each attribute give the player a sense of understanding the upgrading process better. An example of games using multipliers this way for weapon upgrades is Dead Nation, which was used as an inspiration for these particular calculations.

Cannon Hit Point per Upgrade Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Updage Level** | **Level 2 Upgrade** | **Level 3 Upgrade** | **Level 4 Upgrade** | **Level 5 Upgrade** |
| **Upgrade Effect on HP** | + 5hp per upgrade | + 10hp per upgrade | +15hp per upgrade | +20hp per upgrade |
| **Max HP Increase due to specific level upgrades (x3 stats)** | +15hp max | +30hp max | +45hp max | + 60hp max |
| **Total HP after upgrading all stats to specific level** | 65hp | 95hp | 140hp | 200hp |

It is important to note that players can upgrade each attribute individually and do not need to reach a particular level with all attributes in order to keep upgrading. As an example, it is possible to build a Cannon with level 5 Chanceiness, level 2 Attackitude and level 1 Precisionation. Thus, the total HP after upgrading all stats to a certain level are only to give us a rough estimate of different Cannon attributes after some upgrades.

These Hit Point increase with each upgrade occurs for a variety of reasons. We felt that enabling players to upgrade an attribute for HP only would cause the game to be unbalanced and too complex. Indeed, players would need to upgrades other aspects of their Cannons, while remembering to increase the hit points to make sure their other investments don’t go to waste on a Cannon that is easy to destroy. This adds an unwelcome level of complexity for casual players. Thus, a way to increase hit points was needed which did not require specific player action. In order to ensure that Cannons do not gain so much hit points as to effectively turn them into super units, we decided on this scheme which awards extra hit points for each level of upgrade purchased. Upgrades to higher levels, which are more expensive, give more hit points along with their respective attribute improvements.

The hit point increases were balanced so that the number of cannonballs required for a Cannon to destroy a Cannon of equivalent level is always constant. This means that a fully upgraded Cannon will destroy another fully upgraded Cannon in 5 cannonballs, the same amount a base Cannon requires to destroy another base Cannon. This way, we ensure that the gameplay pace does not dramatically change and that gameplay feels consistent. This of course means that the higher level Cannons will destroy the base Cannons faster.

Cannon Value based on Hit Points

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **HP** | **50** | **51-65** | **66-81** | **82-95** | **96-118** | **119-140** | **141-160** | **161-180** | **181-200** |
| **Value** | 100 | 150 | 200 | 300 | 400 | 500 | 600 | 700 | 800 |

Given that Cannons do not have a level per say, but more a combination of levels for each attribute, the way to calculate the Cannon value upon destruction is tricky. This value needs to take into account how tough an opponent a Cannon is, as well as how much the owner of the Cannon spent to upgrade it. One way we found to do this is to give a value based on the hit points of a Cannon. Since the hit points increase with each upgrade, they give a good reference as to the general level of a Cannon. This could have also been done by counting the number of upgrades done of each level on a Cannon, calculating the total cost and defining an acceptable value.

Upgrade Costs in Gold per Level

|  |  |
| --- | --- |
| **Level** | **Upgrade Cost** |
| **Level 2** | 200 |
| **Level 3** | 300 |
| **Level 4** | 500 |
| **Level 5** | 800 |

This table defines the cost of the upgrades per level. It was defined through playtesting based on the value of the Cannons and the benefit gained with upgrading.

**Towers**

Towers are still in development as they were a secondary aspect of gameplay and had been targeted for possible scoping out. The following estimations are still untested.

Please note that the health  and value of towers is still undefined due to the scoping out of towers from the build of the game for submission.

Tower Upgrade Effects per Level

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Level** | **Damage Buff Multiplier** | **+ dmg** | **Healing Multiplier** | **+ hp** | **Shielding Multiplier** | **hp- radius** | **Anti-Air Multiplier** | **r** |
| **Level 1** | (base - 10) | 10 | (base - +5hp/sec) | 5hp/s | (base - 40 hp over a 64 pixel radius) | 40hp-64 pixel | (base - 1shot/3sec) | 32 |
| **Level 2** | X 2 | 20 | X 2 | 10hp/s | X 2 | 80hp - 70 pixel | 2X (1 shot / 1.5 sec) | 40 |
| **Level 3** | X 3 | 30 | X 3 | 15hp/sec | X 3 | 120hp - 75 pixel | 3 X( 1shot/1 sec) | 45 |

As clearly demonstrated by this table, there are 4 types of towers planned for Cannon in C#. The first tower is the Damage Buff tower which will increase the damage of neighbouring Cannons depending on its level. Secondly, is the Healing tower which heals the 4 Cannons adjacent to it by a certain amount of HP/sec depending on its level. The third tower is the Shielding tower, which acts as an umbrella/area of effect shield. The shield strength and area of effect vary according to its level. Finally, the Anti-Air tower will shoot cannonballs out of the air, at a rate specified by its level. The area of effect of this Anti-Air tower is also affected by its level.

Tower Upgrade Costs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Level** | **Damage Buff** | **Healing** | **Shielding** | **Anti-Air** |
| **Base Cost** | 2000 | 1800 | 1800 | 2200 |
| **Level 2 Upgrade** | 2000 | 1800 | 2000 | 2500 |
| **Level 3 Upgrade** | 2500 | 2200 | 2200 | 2800 |

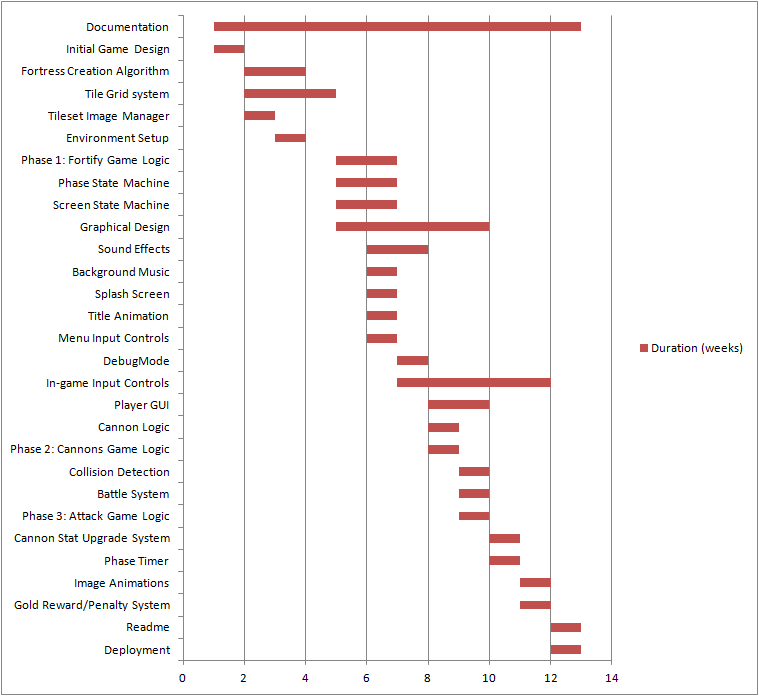
These are the costs of the towers as per original estimate and adjusted by the amount of gold available to players during a normal game as seen during playtesting. Towers are designed to be expensive to purchase and upgrade, because they give special strategic advantages to players. However, they are still affordable to balance out the fact that they are using precious battlefield space which could be otherwise occupied by Cannons.

### Reward Cycles

Cannon in C# is a fast paced game, requiring decision making every second for placing tiles, Cannons, upgrading or fighting. Although it is hard to define which decisions are trivial, minor or major as all actions affect the outcome of the game slightly and no single decision can define the end game outcome, we feel that players will be engaged through this continuous need to act and react to the game and to the opponent.   
  
 Finally, Cannon in C# overlaps the immediate and long term goals, creating a more complex and engaging game. While players will surely begin learning to play Cannon in C# by focusing on each phase individually, on rebuilding their castle, and destroying enemy targets, they will learn to pay more attention to how score varies with their actions and how they can keep track of their long term goal of getting the highest score. These layers of different goals will keep the game engaging and fun to play from the 1st game to the 100th.

# Schedule

The Gantt chart below describes our development schedule. Within the chart is each task that was completed during our development cycle, represented by a solid line denoting the start and end of the task. Our development cycle lasted a total of twelve weeks. Milestones reached coincide with our initial prototype release and final release, at week seven and twelve respectively.



### Developer Tasks

**Roxanne Bériault**Programming, Testing, Documentation, Art Director

**Jonathan Casquinha**  
Graphics Design, Writing

**Adamo Maiorano**Programming, Architectural Design  
**J.B. Phuong-Anh-Vu Lai**Programming, Sound Director

Special Thanks to Linda Komsic for outstanding the art!

# Budget

For this student project, we did not feel that budget was a relevant issue. No money was spent purchasing material or licences.

# Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Date Modified | Revision # | Content Modified | Modifed by |
| 12.06.11 | 2.5 | Added Software Architecture section | AM |
| 12.06.11 | 2.4 | Added Tags & Dialogue Section | AM |
| 12.05.11 | 2.3 | Added Executive Summary and Overview | JC |
| 12.01.11 | 2.2 | Added Characters | RB, JC |
| 11.26.11 | 2.1 | Added Art Direction section | RB |
| 11.25.11 | 2.0 | Added UI Storyboard section from previous notes | JC |
| 11.19.11 | 1.9 | Added Mech. Analysis Section | RB |
| 11.18.11 | 1.8 | Added Rampart and NetStorm Reviews | RB |
| 11.17.11 | 1.7 | Added World Section | RB |
| 11.11.11 | 1.6 | Added Worms, Monkey Island and Tetris reviews | JC |
| 11.11.11 | 1.5 | Added Technology Plan | RB |
| 11.11.11 | 1.4 | Added Player Composite Section | RB |
| 11.03.11 | 1.3 | Added Scheduling  and Budget sections | JC, RB |
| 11.02.11 | 1.2 | Added Progression Graph Section | RB |
| 11.01.11 | 1.1 | Added Controls, Level Design Sections | RB |
| 10.31.11 | 1.0 | Created Template for Design Document | RB |