Gravity

Group 10

Daniel Walz

Jesper Ekhall

Lukas Kalinski

Fredrik Nordh

Alexander Nyberg

User Requirements Definition

Functional Requirements

The Ben use case

Ben is tired after a hard day at work, and is looking for some light and easy entertainment. He has been playing a simple game called Gravity now and then when he has had some time left. In Gravity Ben controls a space ship. Ben is maneuvering his ship in a space inspired playing field. In the playing field there are planets which have a gravity. The gravity of the planets affect Ben's ship. Ben thinks the gravity in the game gives the right challenge and is an important reason for why Ben thinks this game is entertaining.

Becuase Ben is alone he plays Gravity in single player mode. When Ben starts the game in single player mode he gets three ships, that is, chances before the game ends. After all three ships are lost, the game ends. In single player mode asteroids come flying in into the playing field from all directions. Ben can lose a ship by being hit by an asteroid or by colliding with a planet. The goal of the game in single player mode is to get as high score as possible. Scores are gained by shooting down asteroids. One score is gained per asteroid.

The Ben and Greg use case

One night Ben has his friend Greg over. They decide to play gravity in two players mode. They sit down in front of the computer and see their different views in split screen mode. Split screen mode means that the screen is divided into two different sections, with each players view in each section. Both Ben and Greg enjoy the challenge of trying shoot down one another. This time Ben chases down Greg first and shoots him down gaining one score. However, Ben loses focus for a while and crashes into a planet and loses that score again. After some time of playing Greg has 18 points and Ben just gets his 20^{th} point. 20 points was configured points limit and Ben is declared as the winner.

Weapons

At any time the player might use any of the ship's two different kinds of weapon if he has ammunition left. The first kind of weapon is laser. It features rapid fire and the shots move fast. These laser shots are not affected by gravity. However, laser shots are not so powerful. To shoot down an asteroid it takes several laser shots. The second kind of weapon is a missile. The missile is a lot more powerful, but is also slower. Missiles are also affected by gravity.

Moving around the the playing field

The playing field is larger than the screen. This means that the player sees only a portion of the playing field. The view (what the player sees) is centered around the player's ship. When the player's ship moves, the view is zoomed out in proportion to the ship's speed. This gives a sense of speed. When the player maneuvers his ship outside of the playing field the playing field appears as if it was wrapped. For example if the player goes outside of the playing field to the right, he appears again on the left side. The ship is controlled by steering left or right and also by using throttle. The player can use the throttle alone or together with at most one of left or right.

Map Choice

Before starting a game session, the user shall be able to choose the map that that session should be played on. If no active map choice is made, the game system should choose one of the available maps. Each map shall be a definition of the world/environment the player finds herself in while playing a game session.

> *Rationale*: The ability to choose between several maps makes the game more varied as a whole, reducing the risk that the player becomes tired of it too soon. In the case that the player doesn't want to choose a map, the game system may choose one either randomly, by following some constraints or simply by using a default map.

Controls Configuration

Before starting a game session, the user(s) shall be able (but not required) to configure the game controls, i.e., what keyboard keys to use for what action in the game. The game shall provide default controls, allowing the user(s) to change them at any time.

> *Rationale*: Even though the default controls should be chosen to satisfy most users' needs, there still will be those that find them inconvenient in some way. Providing the ability to choose controls will satisfy these users.

Quick Start Help

A user who doesn't know what the game is about and/or what the controls are, shall be able to get a short summary on these points before starting a game session. The summary shall contain the goals of the game play, together with the currently set controls.

> *Rationale*: Providing a quick start help will enable the actual game play to make a quicker impression on the user, as she doesn't have to struggle and “learn by mistake”. The quick start help may also be provided to the user while playing, in case she forgot something.

Single Player or Two Players Choice

Before starting a game session, the user shall be requested to choose whether she wants to play in single player mode or to split up the screen, enabling two users to play against each other. In the case of a “two players” choice, it shall be possible to set controls for and names of both players.

> *Rationale*: Providing both single player and two players modes widens the contexts within which the game can be played.

Two Players Game Rule Choice

Before starting a game session in two players mode, the users shall be able to set a points

limit, defining when the game is going to end. The game shall end when the set point limit is reached by any of the two players.

> *Rationale*: Setting a "game end" rule like this is a convenient way for the users to restrict their playing time, and on a larger scale it'll prevent that they eventually remove the game as a result of uncontrollable (addictive) playing.

Exiting The Game

It shall be possible to exit the game at any stage. When in main menu, the user shall be able to quit to the operating system. While playing, the user shall be able to choose whether to exit to main menu or to the operating system.

> *Rationale*: Being able to exit a game is a fundamental feature, which simply enables the user to decide when she wants to exit and to do so properly.

Sound Effects

Sound effects shall be played for each of the following events: collisions, fired weapons, ship throttle and item pickups.

> *Rationale*: Adding sound effects to this game will make it much more alive.

Single Player High Score List

Exiting the game, either as a result of loosing all ships (game over) or by exiting the game deliberately, shall let the player know about the current high score list and, if she had qualified for a placement on it, request her name. Only exit options provided by the game shall follow this requirement.

> *Rationale*: Having a high score list will give a more "permanent" goal to a game session, motivating the player to strive for a concrete result.

World Boundary Wrapping

When a player makes her ship to go beyond one of the world boundaries, it shall be "teleported" to the opposite side of the world. For example, going out on the left side shall result in appearing on the right side.

Rationale: As the alternative was to “lock in” the ships in the world, which would result in a restricted area of movement, the idea of simulating open space (which in fact is what the game world is supposed to represent) will give the user more freedom about which paths she would like to follow.

Player's World View

The player shall see her ship from above, at a distance that depends on the ship's movement speed. When the speed is increasing, the viewing distance shall increase too. Conversely, when the speed is decreasing, the viewing distance shall decrease too.

Rationale: Providing a speed dependent viewing distance will allow the player to see possible obstacles in time when moving fast, and to see more detail and therefore gain some game feeling when moving at a lower speed.

Single Player Ship Disposal (Lives)

The player shall start having three ships at her disposal. Gaining a certain amount of points shall give another ship. The ships shall not be used simultaneously, but once one is crashed it shall be replaced with a new one if available, otherwise the game ends the achieved points shall be viewed together with a high score list.

Rationale: Restricting the number of ships to expend allows the game to end at a certain point, which is preferable due to the aims of this game – to be a time killer or short entertainment, not a long never ending game session.

Scoring in Single Player Mode

Scores shall be gained partly by shooting at asteroids and partly by the time the player managed to stay alive, having three ships (and therefore chances) at her disposal.

Rationale: As the main challenge of playing in single player mode is to avoid collisions, fire down as many asteroids as possible and stay alive as long as possible, this is a matching scoring strategy, rewarding the player for taking care of these challenges.

Scoring in Two Players Mode

A player shall get rewarded when destroying his opponent's ship with any weapon. A player shall be punished when his ship is destroyed by crashing into some obstacle in the world (including the opponent's ship).

Rationale: Getting scores for killing the opponent and losing scores when “suicide” crashing is a reasonable scoring strategy. Conversely, getting scores for killing and losing scores *when killed*, could, in case of equally experienced players, result in a gaming session lasting forever, which would be in conflict with the purpose of the scoring feature (which is to reduce the gaming time to some extent).

Ship Speed Restriction

A ship's movement shall be restricted in speed. When the speed reaches a set limit, throttling shall not be able to increase it.

Rationale: Not having a speed limit for the ship would result in uncontrollable speed and therefore less fun game play. A speed limit will allow the player to have throttle on all the time, without risking to lose control.

Ship Fuel Restriction

A ship shall either have a fuel restriction (when in single player mode) or have an infinite fuel supply (when in two players mode). When there is a fuel restriction, it shall also be a restriction on how much fuel the ship can have at once.

Rationale: In single player, the fuel restriction will add some challenge in complement to the shooting at and avoidance of asteroids. However, in two players mode the fuel restriction would be rather inconvenient for the player, and therefore it is sufficient with (and even without) asteroids there, as there are two human players playing against each other, having that battle as their primary goal.

Ship Damage

A ship shall be completely damaged when colliding with other ships, a planet or an asteroid. A ship's damage resulting from a weapon projectile hit shall be defined by the destructive power of that projectile.

Rationale: It is reasonable to consider each non-projectile collision as completely destructive for the ship, as the objects that are being collided with are much more solid than a ship. Conversely, a projectile does not always have to completely destroy a ship when hitting it, at least if not being very powerful. Allowing ships to be damaged forces the player to avoid both obstacles and, when in two players mode, the opponent's weapon projectiles.

Operating a Ship

A ship shall be controllable by throttling (i.e., gaining speed in the direction of the ship) and steering right and left respectively. Once a ship's movement and speed is achieved it shall remain constant until its destruction, unless affected by a gravity or its throttling.

> *Rationale*: These controls are sufficient to control a ship. Adding a brake control could be an option, although as we're in space, no such thing should exist. Simulating brakes can be done by turning the ship half a turn and throttling. The ability to control a ship is a basic and expected functionality of this game and therefore it should be provided.

Ship Laser Gun

A ship shall be able to fire laser projectiles. The laser projectiles shall not be affected by planetary gravities. The damage caused by a laser shall be partial. Lasers shall travel fast (in relation to missiles).

> *Rationale*: Since lasers are not affected by gravities, it will be much easier (than with missiles) to hit a target when using them, and therefore their damage power should be reduced. The exact level of damage a laser projectile can cause should be defined at a later stage. The laser gun will enable the player to hit his targets more easily.

Ship Missile Launcher

A ship shall be able to fire missile projectiles. The missile projectiles shall be affected by planetary gravities. The damage caused by a missile shall be complete. Missiles shall be slower than a laser projectile, but faster than a ship.

> *Rationale*: Since missiles are affected by gravities, it will be hard to hit a target when using them, and therefore their damage power should be complete. The missile launcher will enable the player to cause complete destruction when succeeding in hitting a destructible target.

Operating a Ship's Weapons

The player shall be able to choose which weapon to use before firing it off. The projectile resulting from firing a ship's weapon shall be set off from the ship's current position and in the current direction of the ship (i.e., not ship movement, but where the ship will strive to go when/if throttling).

> *Rationale*: Firing a weapon in the direction of the ship is the most natural thing to do, as this is the expected behavior in this kind of games. The player will therefore not have to control both the ship and the ship's guns, but will be able to control both at the same time, making the game play more simple.

Planets

A planet shall have a gravity which shall affect ships, missiles and asteroids exclusively. A planet shall not move in any way. An object being affected by a planet's gravity shall be pulled towards that planet with a certain strength. Asteroids hitting a planet shall, if that is the rule of the game mode or the map, increase the planet's gravitation.

> *Rationale*: Having planets with corresponding gravities makes these more than just simple obstacles to avoid, resulting in a more challenging game play.

Planet Gravity Increase

In single player mode, a planet's mass and therefore gravity shall increase by a certain amount for each asteroid hitting it. In two players mode, this behavior may exist, but it shall depend on how the map is designed. When a planet's mass increases to a certain amount it shall become a black hole, resulting in game over, regardless of the game mode.

> *Rationale*: Planet gravity increase for each asteroid hitting it will be another reason for the player to shoot down asteroids, as well as it will make the game session restricted in time.

Asteroids

In single player mode, asteroids shall be sent in to the world at an adequate frequency, making the game play challenging enough. In two players mode, asteroids may be sent in at a deliberate frequency. Asteroids shall be destructible, both partially and completely. Partial destruction means that an asteroid is split into two smaller asteroids, while complete destruction means that the whole asteroid is destroyed.

> *Rationale*: Having asteroids is most crucial in single player mode, as these, together with increasing gravities (as a result of an asteroid hitting a planet) and picking up fuel items, will be the only challenge for the player.

Items

In single player mode, items containing fuel shall occur randomly in both place and time in the world, and frequently enough to guarantee that the player has a fair chance to pick them up before running out of fuel. In two players mode, items shall not contain fuel, but instead they shall contain weaponry and ship health upgrades. Items shall appear in free space in the world, allowing a player to pick them up.

Rationale: In single player mode, items will force the player to not stay in “safe places” (i.e., where no asteroids hit) but to constantly search for more fuel and therefore actively participate in the game play. In two players mode, items may add some value to the game, by complementing it with unpredictable ship and weaponry upgrades. The usage of an item may be constrained by either time or amount, especially when it makes the owning player more or less invincible.

System non-functional

C++
Central parts of the system depends on the use of object oriented C++ and it shall be used as the only programmering language for this software project.

OpenGL
The user interface shall be in 2 dimensional space and implemented using the OpenGL graphics library.

Sound
The sound system shall use the FMOD API.

Windows XP
The system must be executable on the Windows XP platform.

24 picture updates per second
The system must be able to update the game window at least 24 times per second on a system with 512 MB of memory with a 1 GHz PC processor.

Installation
Installation shall be very simple. The system shall not require to be shipped as an executable installation file but be contained in a zipped file then unzipped to a directory and runable from there without any further requirement.

Write access to file system
The software package requires read and write access to the file system, in particular the directory that the game resides.

Easy too use
It must take an average person within the demographic no more than 15 minutes to be able to use the game.

Maximum hard-drive usage
The game shall not take more than 50MB of hard-drive space.

Development process
In the development process we shall use the trac software project management tool, the SVN source code version control system and the default compiler shall be ????

System functional:

Map loading

The game environment shall be set up by loading a text file from disk, constituting the game world that the game is to be played in. It shall specify the number of planets, their positions, the ships

starting position(s) and how many and how fast the asteroids in the world are.
The map is of the following scheme (<desc> is a number with the description 'desc'):

<Number of planets>
For each planet: <Position X> <Position Y> <Radius> <Mass>
<Max number of asteroids on screen at any time>
<Asteroid arrivals per minute>
<Average velocity of asteroid>

After parsing the input text file the system shall create a world environment with the corresponding parameters to those that the map has given.

World Environment
The world environment shall keep track of every object in the current game world. The objects set shall be constituted by ships, planets, items and asteroids.

Controls Configuration
The system shall store the ship controls specification in a file on disk, residing in the same directory as the game.
The file shall store a keyboard key code for each available control.
The file layout shall be as follows (<desc> is a number with description 'desc'):

<player 1 throttle> <player 1 turn left> <player 1 turn right> <player 1 fire> <player 1 cycle weapon>
<player 2 throttle> <player 2 turn left> <player 2 turn right> <player 2 fire> <player 2 cycle weapon>

Control Handling

The system shall be able to handle several simultaneously pressed and/or held in keys and take appropriate action for each key.

The player uses the controls system and the configuration given in Controls configuration to steer the ship. The throttle key make the ship move straight forward in the direction of the nose of the ship. The turn left and right keys makes the ship steer left and right. The fire key makes the ship fire the currently selected weapon in the direction of the nose.

In the case of the player triggering "turn left" and "turn right" controls simultaneously, the system shall only consider the one which was registered by the system in the first place.

World Boundary Teleportation
When a movable game object reaches the boundary of the world, the collision detector shall tell the world environment about this. The world environment shall then move ("teleport") that object to the opposite side of the world. More specifically, if an object reaches the boundary of axis Y it shall be moved to the other extreme end of axis Y.

Game World Object
An abstract object that encapsulates the standard properties of objects that appear on the screen. These have the following properties:
Position - A vector describing the two dimensional position of the center of the object in the world.
Radius - A floating point approximating the circular radius of the object, used for collision

detection.
Mass - A floating point describing the mass of the object.
Force - A two dimensional vector describing the force acting on the object.
Stationary - A Boolean that tells if this object is affected by the gravity of other objects.

All objects such as ship, asteroids, planets, item boxes and weapons share these properties and shall inherit and implement from this object.

Game World Environment
The full description of the world where the game is played. It consists of and maintains a collection of game world objects. It is responsible for calling the gravitation and collision detection engines.

The game world view must provides services to insert new game world objects and see if the insertion is valid, that is, no two game world objects are allowed to collide at insertion point. Game world objects must also be removable, primarily targeting the need of removing destroyed objects from memory.

Scoring System
The scoring system shall keep track of all player-controlled ships together with events that may affect their score. The scoring system shall also tell the system when a certain scoring limit is met, if any was set, resulting in the end of the game session (this shall only apply to multi player mode). The actual score shall be represented by an integer (both positive and negative) and start at zero. In single player mode the player shall gain one score point for each asteroid it manages to destroy. In two players mode a player shall get one point when killing the opponent with a weapon, and lose one point when killing herself by crashing into something that's not the other player's weapon projectile.

Ship object
The object describing the player controllable ship, inherited from game world object.

It must contain the following attributes:
Speed restriction - An integer limiting the maximum euclidean product of the force of the ships underlying game control object
Fuel restriction - An integer for the amount of fuel carried by the ship. In single player mode using thrust decreases the amount of fuel until the tank is empty and the ship cannot thrust any more. In multi player mode the fuel is unlimited.
Health - An integer describing the health of a ship, starting at 100. When the health drops to or below zero the ship is destroyed.
Picked up Items - What items the ship has collected.

Weapon Projectile Object
An object describing the different weapon projectiles which may be fired by a ship, inherited from the game world object.
Weapon projectiles are laser gun rays and missiles.

It must contain the following attributes:
Impact - An integer describing the strength of the weapon. Higher impact means more damage if hit.

Weapon choice
The player shall be able to cycle weapons using the keyboard. The cycling process will proceed as

follows: the user presses the cycle key and the game displays which weapon is currently equipped, if the user presses the fire button a projectile corresponding to the selected weapon shall be fired. The in game weaponry should consist of at least the following; missiles and lasers.

Asteroids/Planets
The asteroids will be instances of the game world object. They shall not be stationary. If the asteroid is hit by a players weapon (that is if the projectile is on or withing the boundary of the asteroid) it will shatter into pieces, there will be three kinds of asteroids; full sized, half size and quarter sized. If a full sized asteroid is hit it will shatter into several half-sized asteroids and if half-sized asteroids are hit they shall shatter into quarter-sized ones. If the quarter sized asteroid is hit it will disintegrate completely. The asteroids shall be affected by the planets gravities and change trajectory accordingly.

The planets shall be stationary world objects. Planets cannot be destroyed by the player. The players ship shall be destroyed if it crashes into the planet (that is if the ship is on or withing the boundary of the planet.) Asteroids can also crash into planets. The projectiles from a players weapons are destroyed against planets but have no effect.

Item Box Object
Small boxes that may appear at any time in the game world. These contain items that may increase a ships fuel supply, contain weapons that introduce new firing possibilities of the ship or give the ship more health.

Collision Detection Engine
An engine that discovers when two game world objects have collided and provides callbacks when a collision has been detected.

Gravitation Engine

In the system every game world object is attracted to other objects with a strength depending upon the mass of the objects. The attraction shall follow Newton's gravitational laws. The gravitation engine updates each game world object's force vector in each time step depending on the position and mass relative to other game world objects.

Split Screen
In two players mode, the game screen shall be split in half and each player shall control the ship residing in one half of the screen.

Weapon Firing

When a player presses the firing button on the controls a projectile will be sent out in a straight direction from the nose of the ship. The projectile will continue forward until it hits an object in the world.