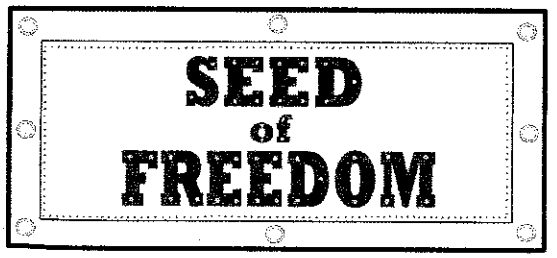
**Scrolling Shooter Game: Seed of Freedom**
**Beta version**




Final Year Project 2

April 2008

By:

Ahmad Zaki Bin Saaid

6518

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

# ABSTRACT

This report are meant to indicates the progress, initial result and available work output done for the Final Year Project (FYP) which is to create a scrolling shooter game for desktop computer platform.

This report includes the objective, the scope of project work, methodology to finish the project within the required time, research done so far and also available result which had been achieved. The report will briefly reviews the project itself; the strategy game itself; within applicable area. That is consisting of general perspective of current strategy games available on the market, its characteristic, and elements to be inserted into the product and current achievements.

The four (5) chapters available cover the introduction, review of related literature, project methodology, results achieved so far and discussion, and lastly the conclusions.

Introduction part discusses the background and trend of scrolling shooter games available currently on the market, problem statement, and objective of the project and scope of works. The second chapter, involve in reviewing the literature review collected within the boundary of the gaming industry. The second chapters mainly tap available review on the internet, documentary done by third party, and conclusion drawn from the writers personal views and experience. The third chapter will evaluate the methodology used to complete the project. They mainly state the working procedure, tools used, familiarization with the tools, and reviews on existing product similar to the project. Fourth chapter, meanwhile, shows achieved result so far and discussions on the item. The fifth and last chapter are going to conclude the whole project briefly and reference used.

At the end of the report, the appendices had been posted to show any diagram, pictures and chart used.

# TABLE OF CONTENT

## LIST OF FIGURES



## LIST OF TABLES

# CHAPTER 1
# INTRODUCTION

## 1.1 Background of Study

Scrolling shooters are type of video game, a subgenre of shoot 'em up (a type of video games that require the player(s) to shoot and destroy as much as possible of appearing enemies) . Such games feature background that appears to scroll as the protagonist moves through them. Scrolling shooters feature 'top-down' or 'side-on' perspectives.

A scrolling shooter is, as the name dictates, a shooter game that takes place against a scrolling background. There are several subclasses of this genre which is horizontal scroll, vertical scroll and hybrid of both.

**Horizontal scroll**

This type views the players' avatar from the side, and presents the level in cross-section, such that the player(s) appears to be flying 'through' the terrain (landscape, spaceship, city etc).

Typically, the scrolling of the terrain in these games is continuous, such that the player is led through each level by the game. There is also sometime a degree of vertical freedom, in which the player can move up or down on playing area which is taller than the screen itself.

As well as battling enemies, some of the challenges in these games tend to come from navigating the terrain, when contact with anything may result damaged or destroy the player avatar.

**Vertical scroll**

Vertical scroll are almost similar to the horizontal scroll, but the direction of scroll and point of view of the camera was taken from different angle: they are viewed from above. This means there is less common solid obstacle from the terrain, as the view shows that the avatar is flying above them. Some people regard this feature to be more challenging with absent of solid obstacle, as the player needs to focus more on shooting and dodging projectiles.

On the other hand, enemies mostly appears from to different planes, which is the "ground" and "air", while the avatar only collide with the enemies from its own plane (ground or air), they still take hits from all enemies regardless of plane. Some of this type also comes with freedom of movement similar to its horizontal counterparts.

**Hybrid**

As specified before, this subtype is a mix of both – horizontal and vertical scroll. Usually a few of existing scroll shooters allow the players to change their view of camera with certain input, while others change their view in different levels (e.g. level 1 to 3 are viewed in vertical while the next level against boss is viewed in horizontal)

### 1.2 Problem Statement

Firstly, the scroll shooter had already born as early as 1980 where an horizontal scroll shooter called *Defender*. Of course there will be a lot more this kind of game that exists in the market nowadays. However, the writer insist on continuing this project for this game with one good reason – limited amount of time to spent on finishing the game.

The targeted player for this game is for busy people especially workers or student with limited times. This kind of people rarely have much time to spent playing games with long game play (such as RPG which could reach up to 60 hours to finish it!). They usually aim at casual games such as puzzle games, card games or flash games. Scrolling shooter like this also favorite for casual players like them.

Most of these people also don't bother to learn new games (especially various online games that appears at present days) as they don't have much time and they also need to focus on their tasks at hand. So, usually they just picked casual games that are easy to learn and play in very short time.

If the view is set from the country point of views itself, the Malaysians had very few game developers to be known globally. Our own citizens had set their own opinions to look at our entertainment-based IT industry to be at lower class compared to foreign games. Painfully the reality is not far from there. Most of the games coming from Japan, United States or European country and they arrival slowly influence the culture of the players (mostly children age 18 and below) wit their own. This mixes of culture could bring benefits but it also threaten our own traditional and local cultures.

### 1.3 Objective of Project

For the objective of the project, this report divide it into two (2) parts, with the first one to answer the problem arose aforementioned earlier in the problem statement, while the second part reveals the strong points of the game itself.

First and foremost, this game is casual game intended for busy people to let the steam out using a short amount of time without neglecting their main tasks at workplace. They could always use this game for a short break just to avoid the stress accumulated burst out on their chair.

This game also aims to be played by people who wish to play easy-to-learn games in short time and finish it as soon as possible. It allow people like parents who is busy but wish to shoot something before continuing their job after a short break.

While this game aim to create a simple game in short amount of time, it also act as the first step for the writer to open his next step into serious programming later on after graduating from the university.

The second part reveals the first of two strong point of the game itself. The writer objective at the end of his game is to create multiplayer scrolling shooter game written using Visual Basic implemented with the ever popular DirectX library. It is created initially by Microsoft to support game development and multimedia programming which come with full support of graphic accelerator, audio support and network functions. Secondly, the game itself is simple enough even for five (5) year old boy to learn in less then one (1) hour – which of course the targeted user, the matured people will finish learn it in five (5) minutes!. Do mind that, the game itself might not be easy to finished in short time though to certain people.

### 1.4 Scope of Work

The scope of this product is generally to dictate the task that needs to be done before, during and after the project start and ended. For such reason, the list is written in point format for easy reading and comprehension by both writer and readers.

1- Study the effectiveness of developing alternative concept among widely used ones; in this project, it is to evaluate the effectiveness of a simple game like this against other casual games.

2- Make sure allow the prototype product meet the prototype plan. Identify any fault and error during the process, if any, before proceed to repair it.

3- Make sure the real product will meet the objective. Identify any lack of characteristic required and make contingency plan to fix the problem.

4- Modify existing scrolling shooter games concept into extra ideas that could be use to promote this product.

5- Observe and identify the changes in perspective of locals towards domestic game development.

6- Identify effects of gaming to the physical and mind health of the users during beta test. Remove any harmful effects that exist and add positive features if applicable.

# CHAPTER 2
# LITERATURE REVIEW

In this part, the report is to show any research done by the project developer, the writers itself, that have little or great influence on the foundation and progress of the project. The content of the literature review is taken from the internet, the writer personal experience and opinion, and also not forgotten, documentary based on military development.

## 2.1 Concept of scrolling shooter

Next, the writer is explaining on the concept implemented on most scroll shooter available on the market. In is divided into weapons, smart bombs, enemies, simplified physics, insurmountable odds, power ups, collision zones and barrages. Please do note that all this is concepts that usually appeared on a typical scroll shooter nowadays and by no guarantee that this entire feat will be implemented in the project.

### Weapons

Being shooters, weapons are one of the most important aspects of these games, and most feature an array for the player to use. Traditionally, the player starts off with only a weak, single-shot, forward-firing gun. This gun can usually be improved or replaced by collecting power-ups.

## Smart bombs

In many vertical scroll (and some horizontal scroll), the player is given a stock of two or three "smart bombs". The bombs are triggered with a button reserved for this purpose and damage or destroy every enemy on screen or in a large area immediately when triggered (with some exceptions, such as *Raiden*). Bombs also clear the screen of enemy fire and usually make the player invulnerable for a very short period of time (usually for the duration of the detonation). Some games reward players with points for each bomb in reserve at the end of a stage or the game. The bombs are generally restocked to the default number when a player re-spawns.

## Enemies

The majority of enemies faced in scrolling shooters is comparable in size to the player character, are quite mobile, and are armed with a small projectile weapon that can fire in any direction; usually at the player's current position. These are basically 'cannon fodder' (its very existence considered only to fill up the area and die consequently later). They are weak, but usually their numbers present a problem. Larger, but mobile enemies tend to have either more guns or more powerful weapons, or both.

Some enemies adhere to surfaces, either by gravity or by clinging to them. These occur in horizontal and vertical scroll; in the latter, however, there is usually little difference between them and airborne enemies, except that touching a ground enemy's sprite is rarely fatal. For example, *Raiden* features tanks that travel along the ground, but they are destroyed in the same way as other enemies. However other games such as *Xevious* and *Deimos Rising* have separate weapon systems to destroy ground targets, usually striking a specific point ahead of the player (to simulate a real-life bomb trajectory).

In horizontal scroll, ground-based enemies present more of a challenge, as they are usually in a difficult position for the player to shoot at, but still fire back.

## Simplified physics

Except in *Thrust*-style games, which are intentionally based on approximated real physics, most scroll ignore physical effects for simplicity (although they may invoke real physics for graphical effect, such as for missiles). In horizontal scroll, for example, gravity is usually completely ignored.

Acceleration and drag are also often disregarded, allowing the player to move with a constant speed, even through denser media such as water. The horizontal scroll *Project X* and *Whip Rush* have been noted for not ignoring the effects of acceleration, which gives them distinctive style and feel.

## Insurmountable odds

Commonly, groups of enemies attack in ordered formations or patterns, known as *attack waves*. One factor mitigating the one-versus-thousands facet is that enemies that are bypassed rarely if ever return to the battle; while no points are scored for their destruction, they no longer attack.

It is very common for scrolling shooters to feature bosses at the end of every level. Often the scrolling stops when the player reaches a boss, others the boss moves at the same speed ahead of the player.

Some shooters also feature 'mini-bosses' - powerful, but lesser bosses - which appear at some intermediate point in the level.

## Power-ups

Power-ups are an integral feature of most "shoots 'em ups". They are enhancements for the player's character that can be gained during the course of the game, usually as a reward for destroying enemies. Power-ups can include speed or armor enhancements, new weapons, weapon upgrades, or autonomous devices that can protect the player and/or add to its firepower. Power-up systems vary from game to game.

A notable example of a scrolling shooter without power-ups is *Radiant Silvergun*. This is not due to primitive design (the game was released in 1998) - the player has three buttons which can be pressed in different combinations to access any of seven different weapons at any time. The weapons are improved in power and range by destroying enemies, not unlike an RPG.

Common non-weapon power-ups include speed boosts and shields.

## Collision Zones

Collision Zones defines areas in which a ship gets hit by weapons released from opposing sides or stage obstacles. As the collision area on the ship (or other player avatar) determines where the player can be "hit," it is often referred to as the "hit box." Before pixel perfect collision detection was feasible, programmers approximated collision zones by using predefined rectangular areas. As computing power increased, pixel perfect collision zones could be calculated automatically, diminishing the need for manually mapped zones.

Some games use a smaller collision zone than the area occupied by the ship's graphics, where a player's ship is destroyed only when an enemy's attack hits the reduced zone, usually located at the cockpit of the ship. This system allows more enemies and shots to fill the screen without making the game impossible to complete. Some shooters even

include smaller collision zones for enemy projectiles, often as small as one pixel in more modern titles (which is often quite necessary).

Taking the concept of smaller collisions further, some games incorporate a two-collision zone system, where hitting only non-critical portions of the player's ship becomes beneficial to the player. This introduces a whole new way to design and play shooters. This system was taken to an extreme in *Psyvariar*, where "buzzing" hits from bullets is essential for high scores, and improves firepower. Some variants of such games have been designed so that it becomes possible to complete the game in a pacifist manner. In other words, you may be able to finish the game without firing a single shot, or at least using only a limited supply of weapons.

## Barrages

Barrages come in all shapes and sizes, but the amount of bullets has increased substantially when shooters started using smaller collision zones. This sub-genre of scrolling shooter is sometimes known as manic shooters, or bullet hell. As barrage patterns became more complicated, game engines began to use interpreters to generate patterns from hardware-independent script files.

**2.2 Military Document**

This game is created based on the World War 2 (WW2) fighter plane where the players need to fight against the Axis air forces to gain air superiority. In this game, nine (9) WW2 fighter plane will be used through out the levels available in the game. They are *Curtiss P-40, P-61 Black Widow, P-63 Kingcobra, F4U Corsair, P-36 Hawk, P-39 Airacobra, P-38 Lightning, P-51 Mustang, and P-47 Thunderbolt.*

The ***Curtiss P-40*** was an American single-engine, single-seat, all-metal fighter and ground attack aircraft that first flew in 1938. It was used in great numbers in World War II. The *P-40* design was a modification of the previous *P-36*; this reduced development time and enabled a rapid entry into production and operational service. When production of the *P-40* ceased in November 1944, 13,738 had been built. They were used by the air forces of 28 nations and remained in front line service until the end of the war.

The *Northrop* ***P-61 Black Widow*** was an American all-metal, twin-engine, twin-boom, monoplane night fighter and night intruder aircraft flown by the United States Army Air Forces during World War II. It was the only Allied purpose-built aircraft to serve as a radar-equipped night fighter. Though the overall shape bears some resemblance to the *Lockheed P-38 Lightning*, this is largely coincidental.

The ***Bell P-63 Kingcobra*** was an American fighter developed in World War II from the *P-39 Airacobra* in an attempt to correct that aircraft's deficiencies. Although the aircraft was not accepted for combat use by the *USAAF*, it was successfully adopted by the Soviet Air Force.

The ***Chance Vought F4U Corsair*** was an American fighter aircraft that saw service in World War II and the Korean War (and in isolated local conflicts). *Goodyear*-built *Corsairs* were designated *FG* and *Brewster*-built aircraft *F3A*. The *Corsair* served in some air forces until the 1960s, following the longest production run of any piston-engine fighter in history (1940 - 1953). During World War II, it was the fighter the

Japanese feared the most. The U.S. Navy counted an 11:1 kill ratio for every *F4U* shot down.

The ***Curtiss P-36 Hawk***, also known as ***Curtiss Hawk Model 75***, was a U.S.-built fighter aircraft of the 1930s. A contemporary of the *Hawker Hurricane* and *Messerschmitt Bf 109*, it was one of the first fighters of the new generation – sleek monoplanes with extensive use of metal in construction and powerful piston engines. Obsolete at the onset of World War II and best known as the predecessor of the *Curtiss P-40*, the *P-36* saw only limited combat with the United States Army Air Forces but was extensively used by the French Air Force and also by British Commonwealth and Chinese air units. Several dozen also fought in the Finnish Air Force against the Soviet Red Air Force. With around 1,000 aircraft built, the P-36 was a major commercial success for Curtiss.

The ***Bell P-39 Airacobra*** was one of the principal American fighter aircraft in service at the start of World War II. Although its mid-engine placement was innovative, the *P-39* design was handicapped by the lack of an efficient turbo-supercharger, limiting it to low-altitude work, although the type was used with great success by the Soviet Air Force. Together with the derivative *P-63 Kingcobra*, these aircraft would be the most successful mass-produced, fixed-wing aircraft manufactured by *Bell*.

The ***Lockheed P-38 Lightning*** was a World War II American fighter aircraft. Developed to a United States Army Air Corps requirement, the *P-38* had distinctive twin booms and a single, central nacelle containing the pilot and armament. The aircraft was used in a number of different roles, including dive bombing, level bombing, ground strafing, photo reconnaissance missions, and extensively as a long-range escort fighter when equipped with droppable fuel tanks under its wings. The *P-38* was used most extensively and successfully in the Pacific Theater of Operations and the China-Burma-India Theater of Operations, where it was flown by the American pilots with the highest number of aerial victories to this date. In the South West Pacific battle, it was a primary fighter of United States Army Air Forces until

the appearance of large numbers of *P-51D Mustangs* toward the end of the war. The *P-38* was the only American fighter aircraft in active production throughout the duration of American involvement in the war, from Pearl Harbor to VJ Day (Victory over Japan Day – literally a day to commemorate the end of WWII).

The *North American Aviation* ***P-51 Mustang*** was an American long-range single-seat fighter aircraft that entered service with Allied air forces in the middle years of World War II. The *P-51* became one of the conflict's most successful and recognizable aircraft. The *P-51* flew most of its wartime missions as a bomber escort in raids over Germany, helping ensure Allied air superiority from early 1944. It also saw limited service against the Japanese in the Pacific War. The *Mustang* began the Korean War as the United Nations' main fighter, but was relegated to a ground attack role when superseded by jet fighters early in the conflict. Nevertheless, it remained in service with some air forces until the early-1980s. As well as being economical to produce, the Mustang was a fast, well-made and highly durable aircraft. The definitive version of the single-seat fighter was powered by the *Packard V-1650-3*, a two-stage two-speed supercharged 12-cylinder *Packard*-built version of the legendary *Rolls-Royce Merlin* engine, and armed with six of the aircraft version of the .50 caliber (12.7 mm) *Browning* machine guns. After World War II and the Korean conflict, many *Mustangs* were converted for civilian use, especially air racing. The Mustang's reputation was such that, in the mid-1960s, Ford Motor Company's Designer John Najjar proposed the name for a new youth-oriented coupe after the fighter

### 2.3 Personal Experience and Opinion

While getting idea and research material from external sources deemed to be extremely resourceful and worthy, the writers also draw such lot from his own opinion and experience as a computer and video games player, or usually called 'gamers', to write this report.

Through out his childhood, the writers had played (as much as he remembers) is three (3) game titles. Those title are *1942*, *Gradius*, and *Stargate*.

*1942* is definitely an addictive, vertical scroll shooters games. It was easy to learn but definitely not easy to win. The game was played in *NES* platform, or most people called the cartridge game from early 90'. It put players into *P-38 Lightning*, the twin booms fighter plane against horde of Japanese air fleet to reach the last destination, Tokyo. It was magnificent at that time (the writer still thinks it is!) with the plane ability to spin on the air to avoid enemy fire. Since the platform doesn't have save functions, the writer never finish the title as the platform power adapter accidentally burned due to long hours of playing.

The second title, *Gradius*, allow the player to control a futuristic fighter plane that fly across space to fight alien beings. The unique points of this game lie with its flashy power-ups such as land missiles, laser, rapid pulse rifle and most interestingly the autonomous blinking sphere with ability to shoot similar weapon as the player avatar. Its graphic is considered extremely good compared to its peers on the *NES* platform.

Finally, the *Stargate*, is an interesting game that use simple physic concept of inertia and momentum of the player spaceship while in combat with the enemies. The enemy, consist of various alien beings aim to abduct humanoids and subsequently merge with them to become more powerful, while at the same time destroying the planets protected by the player.

More review of this game will be written in the later section of this report.

# CHAPTER 3

# METHODOLOGY

## 3.1 Procedure

### 3.1.1 Research procedure

**Individual research time**

- In this period, the writer spends around 6-7 hours per week to research any necessary data to complete the product.

- The search data consist of:

a) Information on the product element. E.g. Nazi fighter plane, Allied fighter plane etc

b) Reference on coding a game product.

c) Gathering necessary open-source library for the product.

**Group discussion**

- A weekly meeting, around 30 minutes, were spend to allow the writer consult the assigned supervisor about the courses and advise on the project.

### 3.1.2 Working Procedure

- 6-10 hours each week were spent by the writer to focus on writing, fixing and modifying the product.

### 3.2 Tools Required

The project will make use multiple types of tools to assist specific area to complete the prototype and product to be delivered. Those tools, listed according to its priority value to the project from top-bottom order, come from four (4) different aspects, programming, 2-D software, sound editing, and music editing.

From the first aspects, programming, the writers decide to use C++ language. Thus the tool selected is *Microsoft Visual Studio 6.0 complete*. This software allow programmers from different levels to complete their task with assisted pre-installed library functions and auto-suggestion functions to help user to use more detailed command line. The previous software used by the writers is *Borland C++*, while it is good for basic usage and cheaper compared to the *Visual Studio*, its functions are lesser and doesn't fully accommodate requirement to advance further into programming game engines.

The second aspect is the 2-D elements. In this aspect, it is to be divided into three (3) different fields, which are drawing, painting, and image processing/editing. Firstly, the drawing task will be finished using *Adobe Illustrator* with assistance of the writer drawing skills and scanner usage. The writers will draw basic figures of game elements before scan it and deliver it to the *Adobe Illustrator* for further enhancement. For cheaper and easier alternative software, *Inkscape*, was put into consideration. For painting job, the writers decide to pick *Paint Shop Pro* for basic pictures coloring. Finally, the painted pictures will be edited and processed using *Adobe Photoshop* into better quality to be used as graphical artworks, game trailer and also game elements such as the interfaces, character models in the game, items, background and character portrait. Since the tools price was totally out of question by normal students, the writers intend to use any method necessary to get the tools required to complete the project within one (1) year of studies that is if no extension of studies required in the future.

The third aspect is the sound editing. Simple as it sounds, the writer intends to use his knowledge on *Audacity* to create necessary sound for the game sound effects. While this require specialized skills and creativity and not yet being venture deeper, the writer will request for external helpers if necessary and such helpers will be discuss in later report if applicable.

The last aspect is the music editing. The music is used for any events that require them especially for background music to bring forth necessary emotions according to the events triggered during the game play. The music might also be needed during the creation of the game trailer. One of the software initially picked is *Cakewalk*. Further details are to be included on later report as the task is yet to be implemented by the writer.

Out of five aspects required to complete the game, the writer dare not to spread his focus but to the first three (3) main aspects, which is the programming, 2-D elements, and 3-D modeling. The fourth and fifth aspects remained little or untouched until it is needed for the future development. Most of the tools such *Adobe* tools were priced extremely high in the market and absolutely unaffordable normally to the students. However, in Malaysia there is always a method to overcome that, and for such opportunity arose, those tools were chosen by the writer.

### 3.3 Familiarize with Software

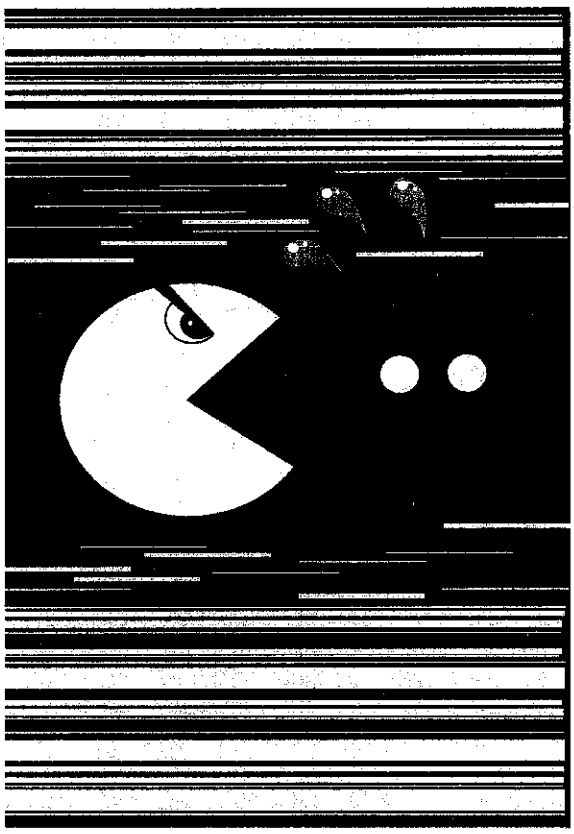
So far the writers had been working to familiarized him with the tools mentioned beforehand especially during the lab of *Multimedia System* course lead by Mr. Yew Kwang Hooi and *Multimedia Programming* led by Mr Nordin.

Tools that had been used by the writer is *Adobe Photoshop*, *Inkscape* (for purpose of creating images) and *Audacity* editing. These works posted after this is few of several artworks finished.


* Figure 1- Multimedia and Virtual Reality poster. Created using *Adobe Photoshop*.


* Figure 2- CD Cover created for multimedia lab using *Inkscape*

* Figure 3- Human faces created using *Inkscape*



* Figure 4- Pictures based on the popular 80's video games, Pac man, created using *Inkscape*


* Figure 5- Greeting cards cover for unknown occasion. Created using *Inkscape*

Another achievement is using *Audacity* software to edit sounds and songs. The writer manages to create a sound clip featuring his voice giving speeches and followed by applause by imaginary audience. It was created by mixing several sound clips such as applause sounds and hall environmental effects with recorded writers' voice. The song edited was *fate stay night - disillusion (Tainaka Sachi)-original.* The song vocal was removed while maintaining the instrumental sound. It was completed after the writer followed a guide from the *Audacity* website.

### 3.4 Reviews on Existing Similar Product

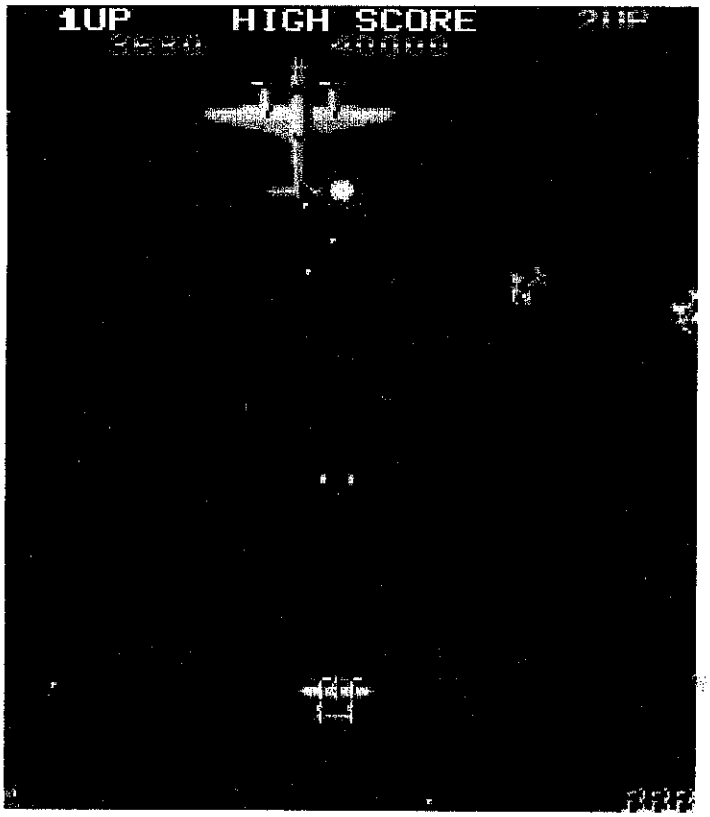
Reviews on existing similar product bring the reader to the three (3) title mentioned earlier *1942, Gradius, and Stargate.* As all this three game, more or less, influence this project idea and so forth, the writer intend to get into a bit details on this three title.

***1942***

***1942*** is a vertically scrolling shoot 'em up made by Capcom that was released for the arcade in 1984. It was later ported to the *NES, MSX, Amstrad CPC, ZX Spectrum, NEC PC-8801, Commodore 64* and *Game Boy Color* (all these are old game platform from late 80's to early 90's). It was included as part of *Capcom Classics Collection* for the *Xbox* and *PlayStation 2* in 2005. *1942* is the first game in the 194x series, followed by *1943: The Battle of Midway.* The NES Version was developed by *Micronics*

*1942* is set in the Asian theater of World War II. Despite the game being created by a Japanese company and staff, the goal is to reach Tokyo and destroy the entire Japanese air fleet. The player pilots a plane (dubbed the "Super Ace", although its appearance is clearly that of a *Lockheed P-38 Lightning*), and has to shoot down enemy planes. Besides shooting, the player can also perform a "loop-the-loop" to avoid enemy fire.

There are 32 levels, the ending of each finishes with the plane landing upon an aircraft carrier, receiving a debriefing and a briefing for the next mission. Players have to travel through *Midway, Marshall, Attu, Rabaul, Leyte, Saipan, Iwo Jima,* and finally *Okinawa* before reaching the ultimate goal, *Tokyo*. On the *Famicom (NES)* version, all enemies still displayed when it lands on the carrier will explode and earn the player points. This is not so in the Arcade version, which makes it very hard to attain 100% rating on level completion. Some stages mark the appearance of the Japanese Mother Bomber **Ayako** (based on an actual Japanese bomber, see *Nakajima G8N*), which must be shot down to complete the stage. Each level increases the difficulty and planes become more aggressive as you move throughout the 32 levels




* Figure 6- a screenshot of the *P-38* during a fight with a mini boss.

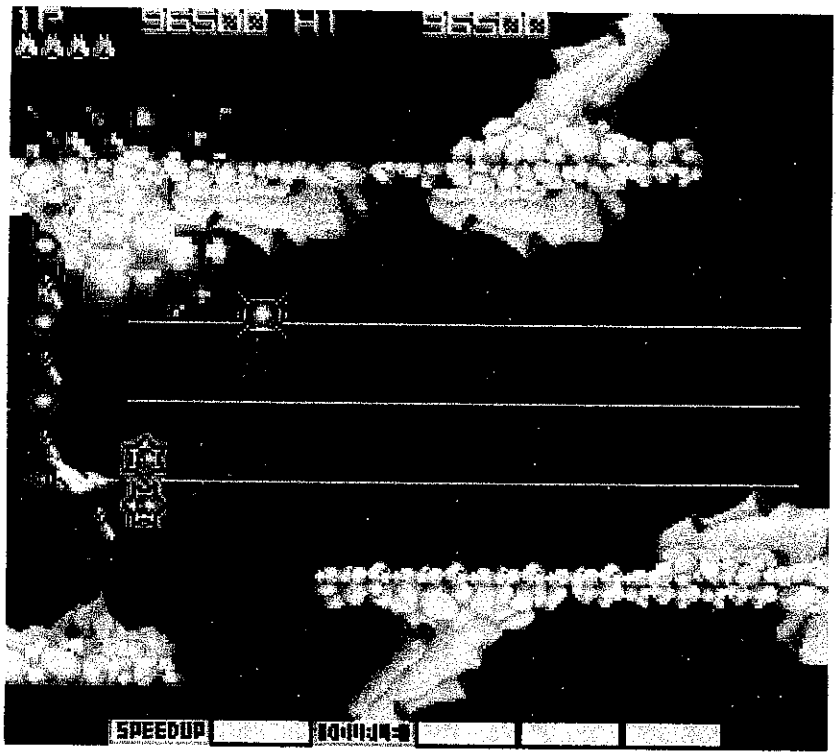
***Gradius***

***Gradius*** is a horizontally-scrolling shoot 'em up released by *Konami* in 1985 for video arcades. It was originally released in North America and Europe as ***Nemesis***, but was re-released as part of the *Gradius Collection* (2006) as ***Gradius*** in these regions.

*Gradius* has the distinction of popularizing a weapon selection bar called "Power meter", based upon collecting capsules to 'purchase' additional weapons. The game was ported to many systems, most notably the *Nintendo Entertainment System – NES* (with this version also appearing on the *Wii's* Virtual Console).

The player controls the trans-dimensional spaceship *Vic Viper*, and must battle waves of enemies through various different environments.

The game became synonymous with the phrase, "Shoot the core!", as the standard of boss battles in the *Gradius* series involved combat with a giant craft, in the center of which would be situated one to several blue colored spheres. These bosses would be designed in such a way that there would be a straight passage from the exterior of the giant craft which leads directly to one of these cores. The player must fire shots into this passage whilst avoiding attack patterns from weapon emplacements on the body of the boss. However, small but destructible walls are situated in this passage, impeding the bullet shots from damaging the core, and must be whittled away by repeated well-placed shots. In a way, these tiny walls represent the boss' shielding gauge until its core is finally vulnerable to attack. Some bosses have the ability to regenerate these walls. When the core has sustained enough hits, it usually changes color from blue to red, indicating that it is in critical condition and its destruction is imminent. Upon the destruction of a core, a piece of the boss may be put out of commission, seeing that it is no longer powered by a core, or if all of the cores are destroyed, the entire boss is defeated and explodes satisfyingly. Note that these cores are not present on the more organic bosses of *Gradius*. Such bosses have weak spots in places such as a mouth, head or eye.




* Figure 7 – The *Vic Viper* is assisted by three (3) autonomous spheres.

***Stargate***

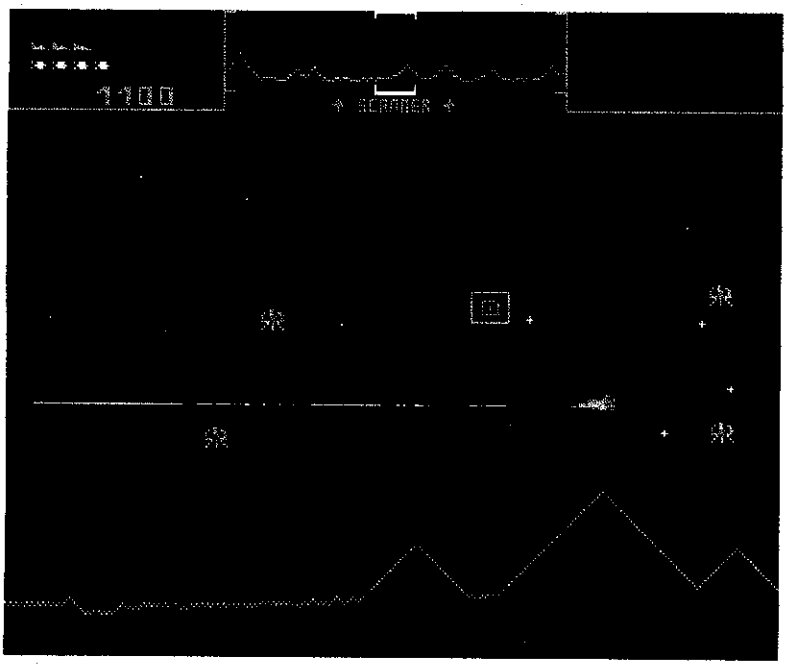
Created by Eugene Jarvis, ***Stargate*** is an arcade game released in 1981 by *Williams Electronics*. It is a sequel to the 1980 game *Defender*, and was the first of only three productions from "Vid Kidz", an independent development house formed by Jarvis and Larry DeMar

The game is also known as "*Defender Stargate*" and "*Defender II.*" The latter name was used in home video game releases, due to legal issues (according to the bonus material for *Midway Arcade Treasures*, Williams wanted to "make sure they could own the trademark" on the "*Defender*" name). The name "*Defender II*" has been used on all of its home ports, and game compilation appearances, however there were never any "*Defender II*" arcade units. To complicate matters, the *Atari 2600* port was originally sold under the "*Stargate*" moniker but was renamed to "*Defender II*" for a later re-release

This sequel adds new enemy ships to the alien fleet such as *firebombers*, *Yllabian Space Guppies* (note that Yllabian is based on "Yllab", the word "Bally" spelled backwards, a friendly poke at Williams' then-competitor, Bally-Midway), *Dynamos* and *Space Hums*. The Defender ship is now equipped with an *Inviso* cloaking device, which renders the ship invulnerable when activated, but has a limited charge. A *Stargate* will transport the ship to any *humanoid* in trouble. There are now two special stages, the *Firebomber Showdown* and the *Yllabian Dogfight*, that occur every fifth and tenth wave

The game is much harder than its predecessor, though world-class players such as Wes Simonds, Bill Fye, and Charles Warrell are all known to have scored over five million under strictly-controlled tournament conditions. As in the first game, if all the humans are captured the planet explodes and turns all the landers into mutants

This video game has no connection to the *Stargate* franchise



* Figure 8 – The swarms of alien surrounded the player avatar.

# CHAPTER 4
# RESULTS AND DISCUSSIONS

## 4.1 Structure of the Game

The structure of the game will be evaluated into several parts. The first part is the game components and then the second part involves game concept details on how the game engines mechanism works.

The first and foremost part is the game components used to visualized process happened from the start of the game until the players exit the program. The flow of the game is initializations process which includes memory allocation, loading necessary files and build tables. Then it will proceed to the main event loop to call windows of the game and initialized timing. From there the players can input any command needed such as giving command to the game characters, interact with the game interfaces and elements, and also exit the game.

If the player decides to exit the program will run cleanup process where memory reallocation and closing files take place. Then the program will exit to the operating system (OS).

If the game continues, the game structure will handle any triggered events and run the main logic of the game. The main logic contains most of the process required to run the program. Next it will process the output and deliver it to the monitor. See the game flowchart at APPENDIX A, PRODUCT FIGURES on Figure – Game Components

The second part revolves around basically how the game should run. After the game being open by the user, it will show the entry screen (welcome screen). From there, it show the mission briefing and the plane player will be using. This process is repeated for each opening of each level.

The player will be required to move the plane from left to right in order to avoid enemy fire while pressing 'shoot' button to destroy the opponents.

The player will fight against two types of enemy, fighter plane and anti-air turret on the ground. The enemy fighter planes are adapted from *Axis* air forces in WWII especially the common German *Luftwaffe* and BF-*109*. Generally all planes will shot volley bullets while the turret use volley of missiles to destroy the player avatar.

Each different fighter plane that available to the player is specified with different attributes. Those attributes are its durability, firepower and mobility. In the prototype, each plane are used in fixed levels respectively which means the player got to use the weaker plane, *P-40*, until later levels where the player got their hands on the strongest plane in the game, *P-47*.

The durability signifies how many punishment the plane can take before being destroyed. The firepower represents the strength of its bullets against the enemy durability. Lastly, the mobility attributes shows how fast the plane can strafe according to player input while avoiding incoming enemy fire.

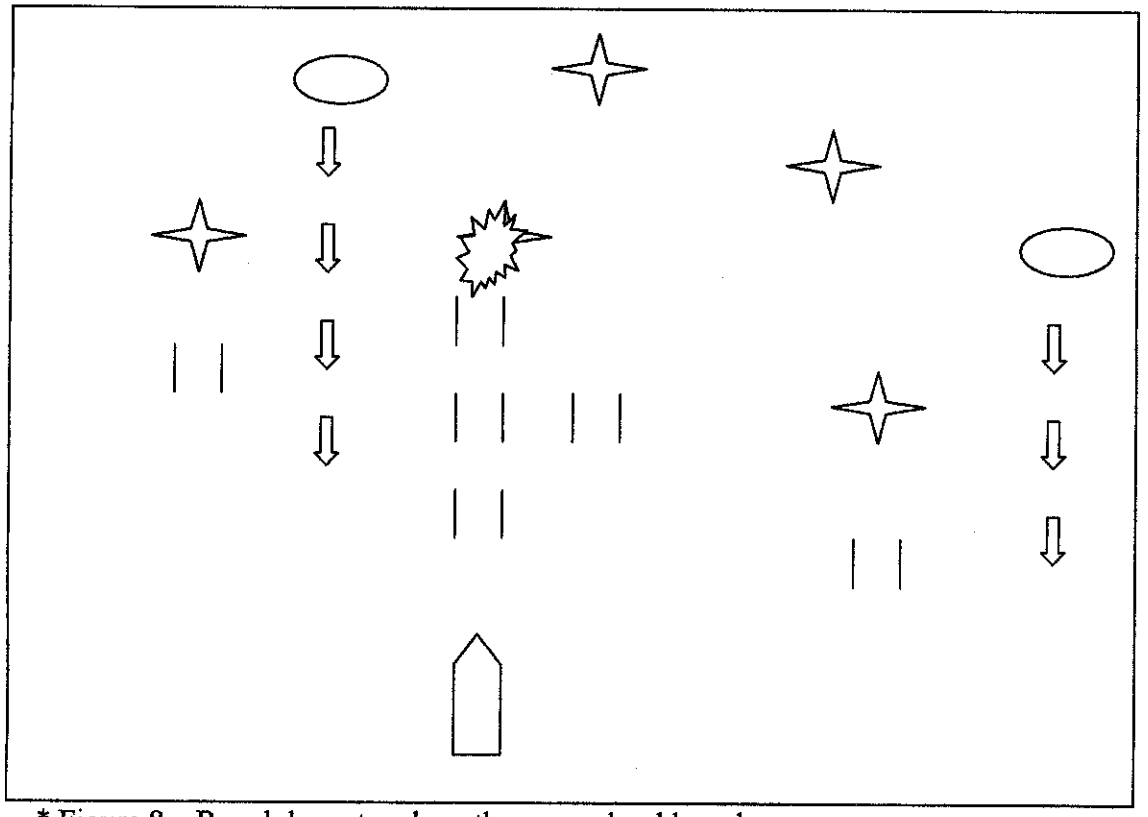
| Plane | Durability | Firepower | Mobility |
|---|---|---|---|
| P-40 | 2 | 4 | 1 |
| P-61 | 3 | 4 | 1 |
| P-63 | 3 | 4 | 1 |
| Corsair | 4 | 4 | 2 |
| P-36 | 4 | 5 | 2 |
| P-39 | 5 | 6 | 2 |
| P-38 | 7 | 7 | 3 |
| P-51 | 7 | 8 | 3 |
| P-47 | 9 | 9 | 3 |
| Luftwaffe (enemy) | 3 | 1 | 1 |
| Luftwaffe sea-variant (enemy) | 3 | 1 | 1 |
| Anti-Air turret (enemy) | 9 | 4 | 0 |
| Gunboat (enemy) | 9 | 4 | 0 |

* Table 1 – Show available unit for the prototype

*Notes: for the purpose of testing the entire player plane durability is multiplied by 100.

### 4.2 Game Element Design

**Game layout**



* Figure 8 – Rough layout on how the game should work

Legend:

= The player plane

= The enemy plane

= Anti-air turret

= Bullets

= Missiles

= Explosions

# CHAPTER 5
# CONCLUSION

While maintaining to continue to fulfill the basic concept of the game, the writer plan to continue more development into the game feature and increase the product marketability. The writer also continues his research on advance game programming (especially on multiplayer programming). This includes planned research on foreign country with historical military forces such as America, Russia and Britain. The intended plan was to hasten the development and completing the game trailer and the game prototype during the midterm holiday that was given to the student.

# REFERENCE

The reference source comes from various forms which are:

a. Reference and guide book for beginner and intermediary level learner.

[1] Lynn T. Harrison, *Introduction to 3D Game Engine Design Using DirectX 9 and C#*, for insights of how the game process works and related issue with DirectX 9.

[2] Andre LaMothe, *Tricks of the Windows, Game Programming Gurus, Fundamentals of 2D and 3D Game Programming*, for tips and guide to create a complete game.

[3] Jan Kabii, *Photoshop 7, Complete Crash Course*, for teaching the writer on using Adobe Photoshop 7.

[4] Jonathan S. Harbour. Visual Basic Programming with DirectX, for basic to advance guide on programming VB with DirectX

b. Official learning material:

[1] Mr Yew Kwang Hooi , *Multimedia System* courses assigned lecturer, in learning various aspects of great multimedia tips and also learning materials during lab session on *Adobe Photoshop, Inkscape,* and *Audacity.*

d. Documentary:

[1] *Modern Marvels - Deadliest Weapons, History Channel*, for giving information on deadly weapon during the past few wars.

[2] Richard Machowicz, *Future Weapon, Discovery Channel*, for revealing United States latest warfare development, and deadly equipment.

# APPENDIX A

# PRODUCT FIGURES

## Product Flowchart

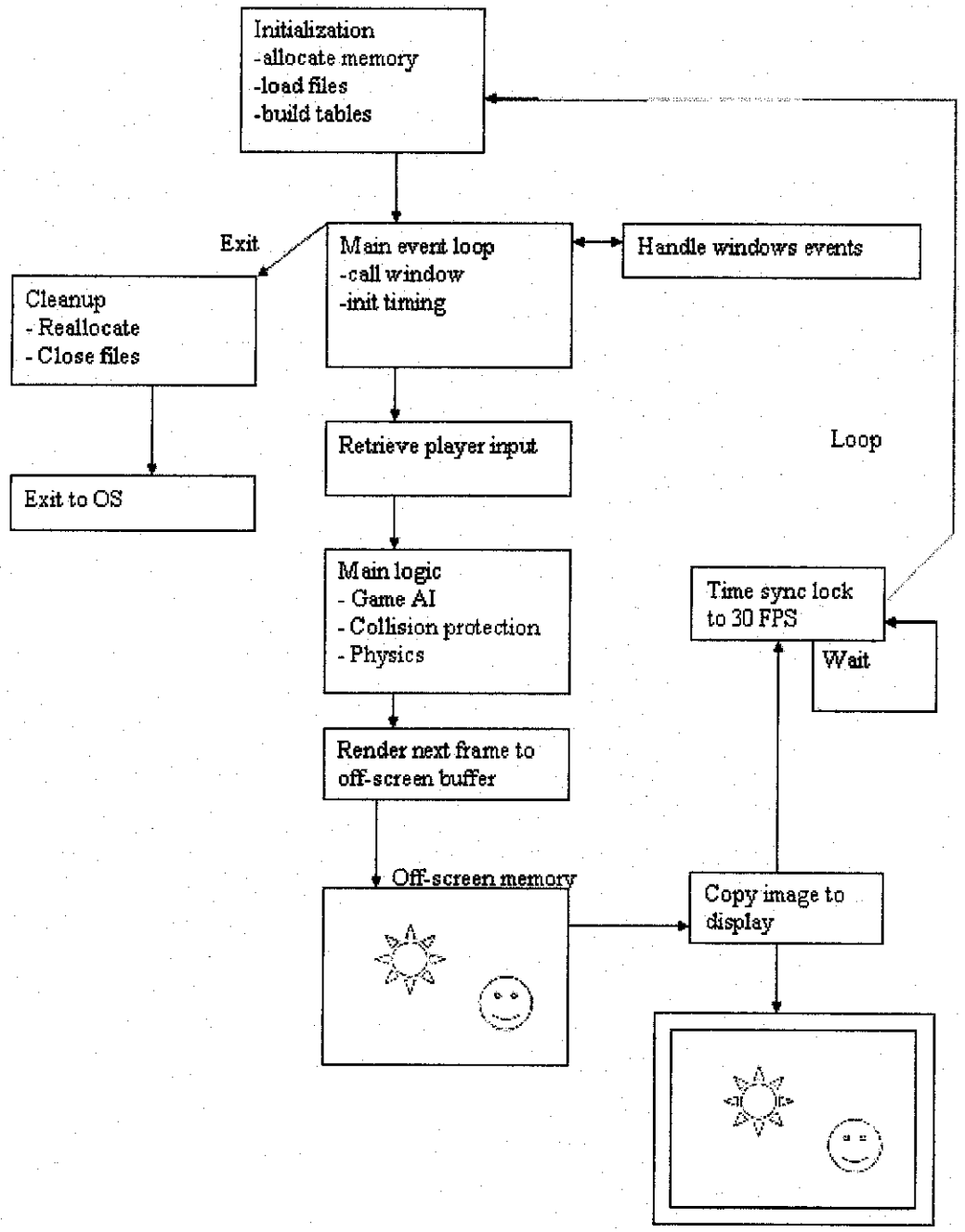



*Figure 15– Game Components

**Product Game Structure**

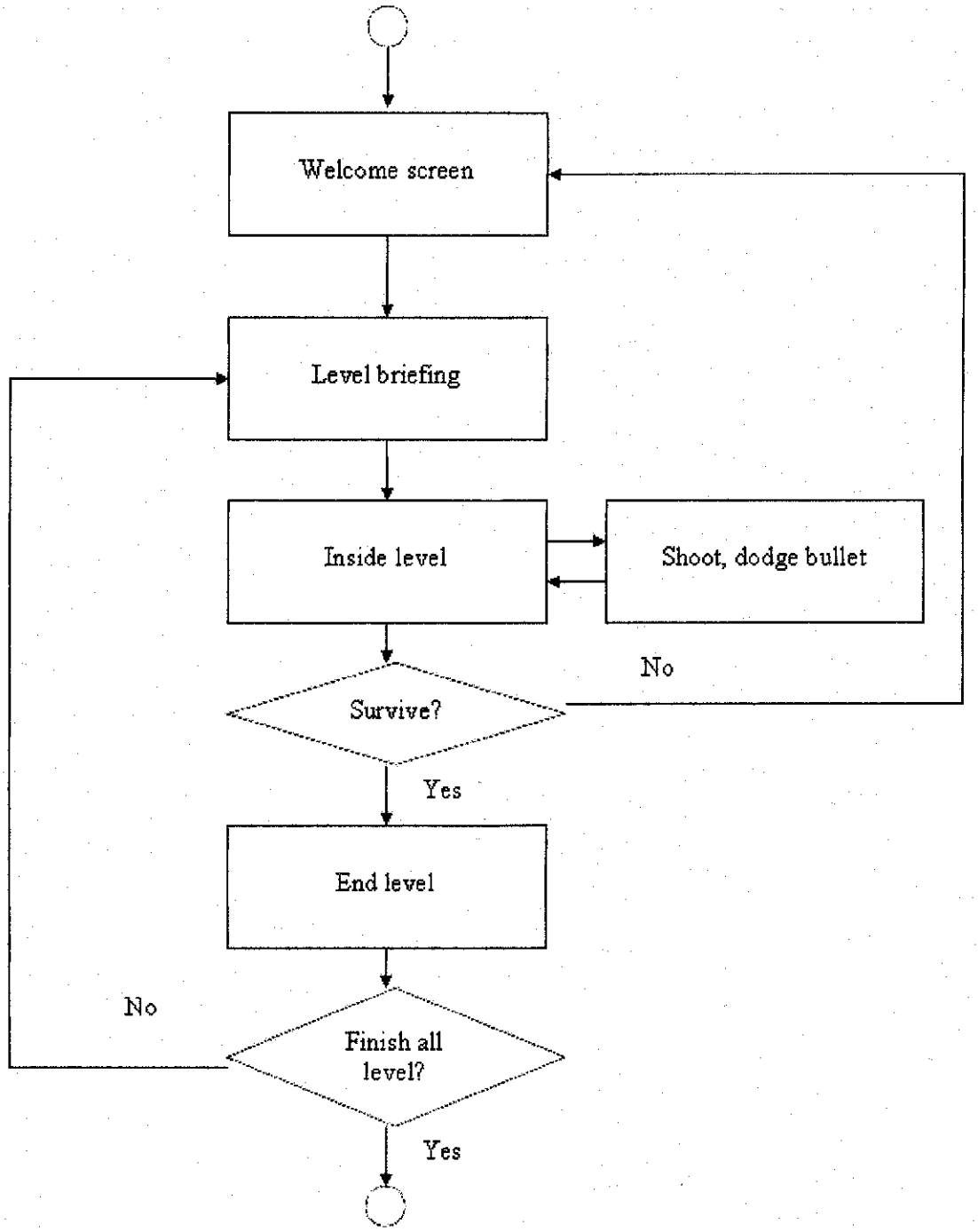
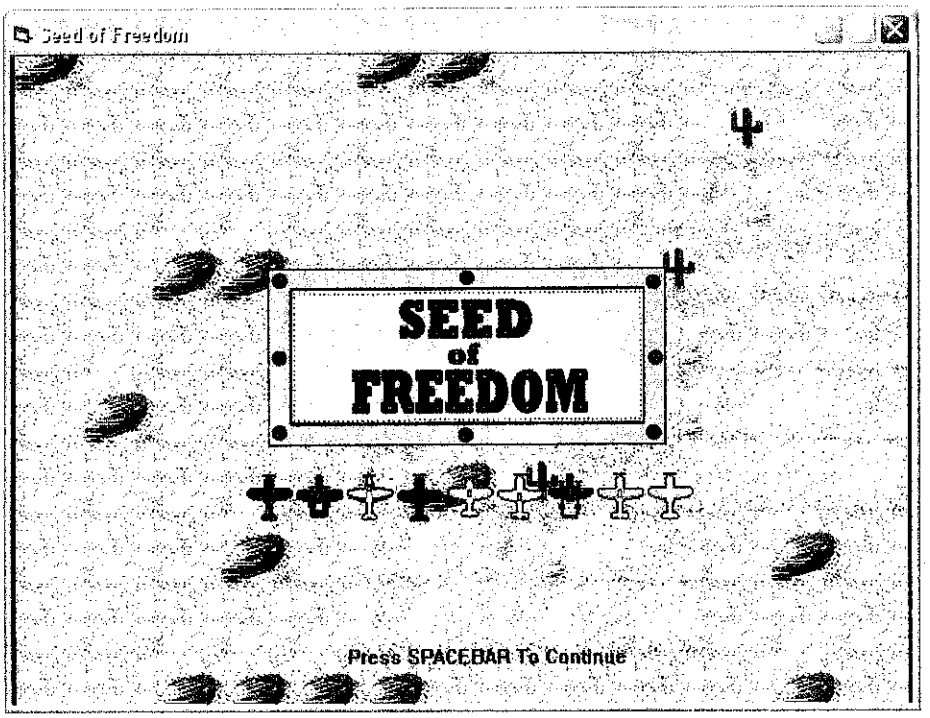
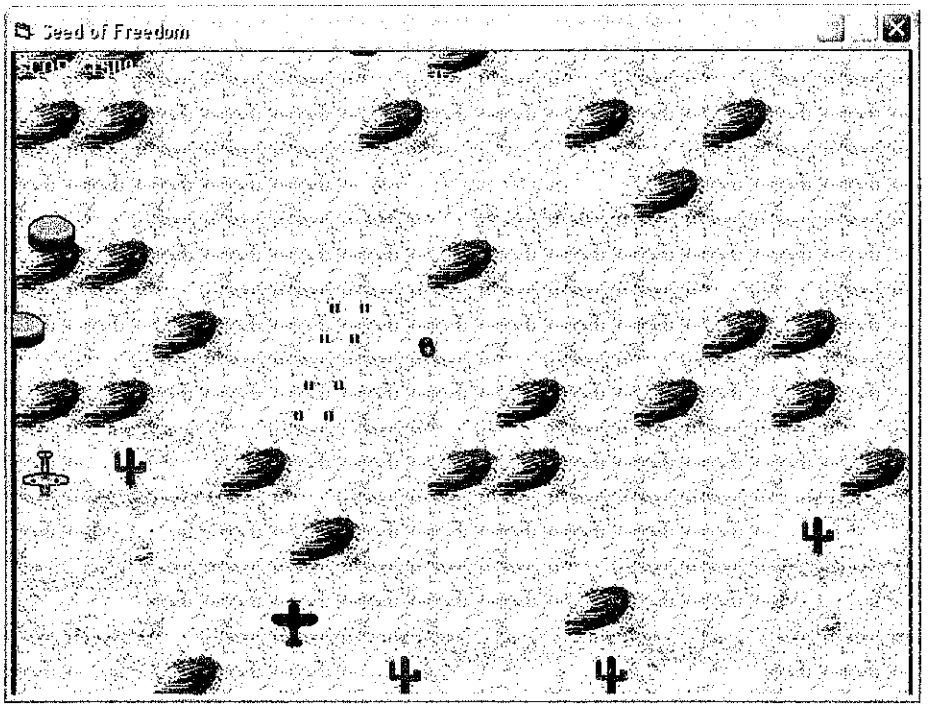



Figure 16- Game structure

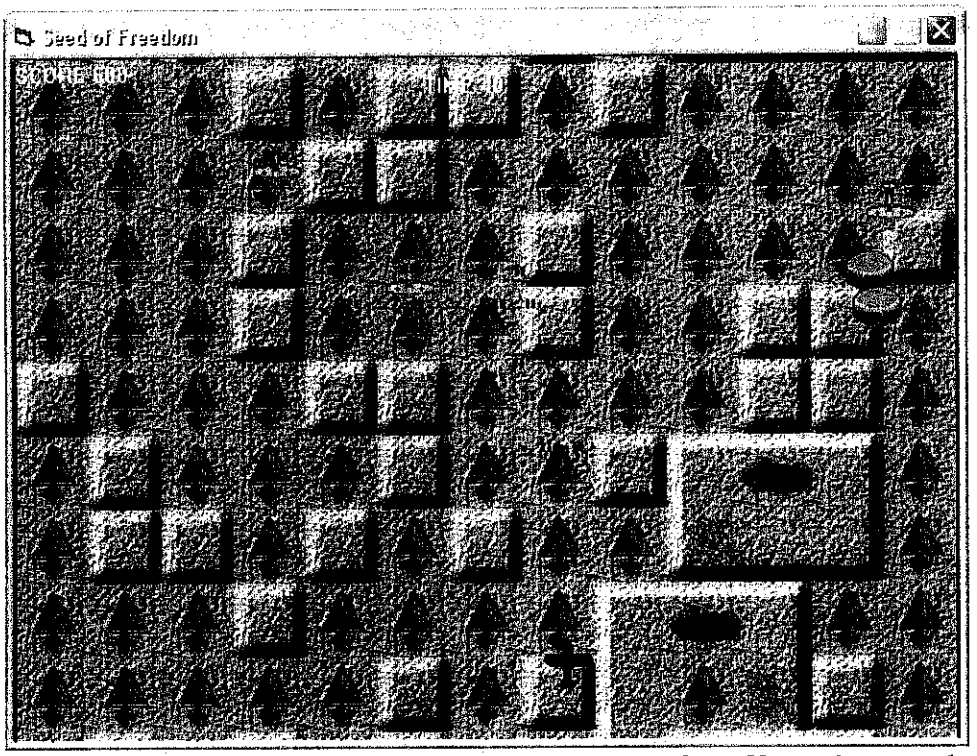
## Product Screenshot



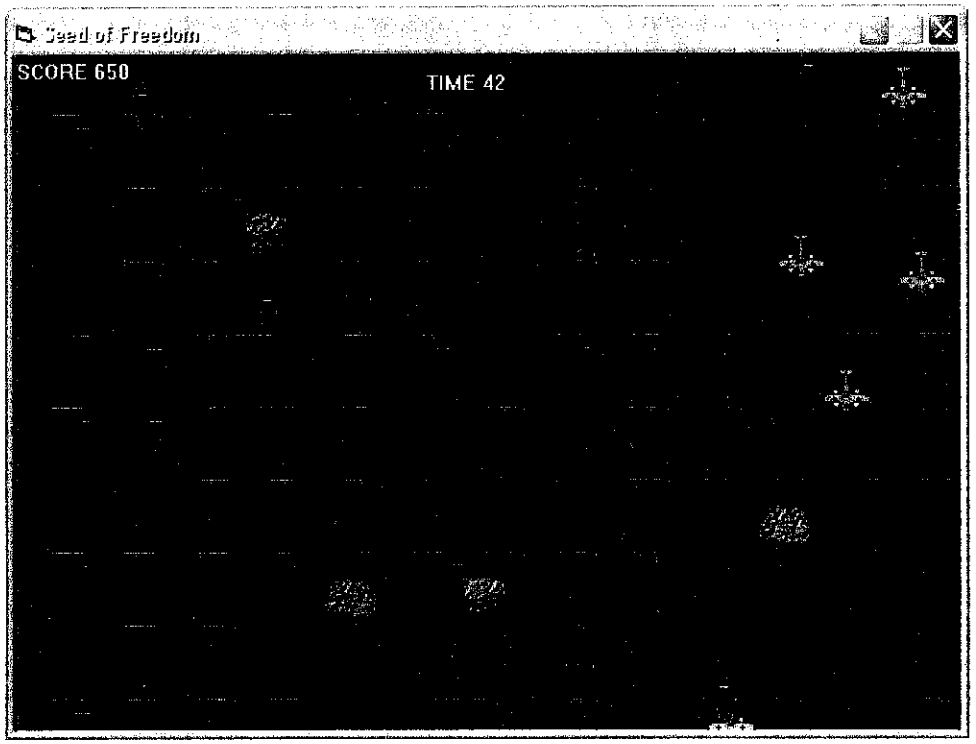

* Figure 17 – Main menu




* Figure 18 – Game play using desert tiles using Luftwaffe and ground turret as the enemy




* Figure 19 – Game play using plain tiles using Luftwaffe and ground turret as the enemy




* Figure 20 – Game play using sea tiles using Luftwaffe sea-variant and ships as the enemy

# Partial Code

```
Option Explicit
Option Base 0

'program constants
Const MAXFRAMERATE As Long = 100
Const MAXBULLETS As Long = 10
Const MAXENEMIES As Long = 30
Const MAXEXPLODES As Long = 10
Const MAXGUNS As Long = 30
Const MAXROCKETS As Long = 10

'DirectInput key codes
Const KEY_LEFT As Long = 203
Const KEY_RIGHT As Long = 205
Const KEY_UP As Long = 200
Const KEY_DOWN As Long = 208
Const KEY_LCTRL As Long = 29
Const KEY_ESC As Long = 1
Const KEY_PGDN As Long = 209
Const KEY_SPACE As Long = 57
Const ALLSTOP As Long = 999

'late-initialized constants
Dim CWHITE As Long
Dim CRED As Long
Dim CGREEN As Long
Dim CBLUE As Long
Dim CBLACK As Long

'program structures
Private Enum GAME_STATUS
   LOADING = 1
   SHOWTITLE = 2
   RUNNING = 3
   NEXTMISSION = 4
   ENDGAME = 5
End Enum

Private Type POSITION
   X As Long
   Y As Long
```

```
End Type

Private Type BOUNDARY
   Left As Long
   Top As Long
   Right As Long
   Bottom As Long
End Type

'program controls
Dim WithEvents Picture1 As PictureBox

'DirectX components
Dim dinput As clsDirectInput8
Dim WithEvents Keyboard As clsDIKeyboard8
Dim WithEvents Joystick As clsDIJoystick8
Dim dsound As clsDirectSound8
Dim dswFire As clsDSWave8
Dim dswBoom As clsDSWave8
Dim dswRocket As clsDSWave8
Dim dmusic As clsDirectMusic8
Dim dmsSong As clsDMSequence8

'game library objects
Dim csPlayerBullets(0 To MAXBULLETS) As clsSprite
Dim csEnemyBullets(0 To MAXBULLETS) As clsSprite
Dim csEnemyRockets(0 To MAXROCKETS) As clsSprite
Dim csEnemy(0 To MAXENEMIES) As clsSprite
Dim csGuns(0 To MAXGUNS) As clsSprite
Dim csExplosions(0 To MAXEXPLODES) As clsSprite
Dim WithEvents Scroller As clsVScroller
Dim cbBackBuf As New clsBitmap
Dim csPlane As New clsSprite
Dim csMission As clsSprite

'variables related to game status
Dim lMissionNumber As Long
Dim GameStatus As GAME_STATUS
Dim bGameOver As Boolean
Dim bMissionLoaded As Boolean
Dim lCurrentMission As Long

'variables related to player
Dim lPlayerHealth As Long
Dim lPlayerDamage As Long
Dim lPlayerSpeed As Long
```

```
Dim lPlayerAttack As Long
Dim lPlayerScore As Long

'variables related to projectiles
Dim lBulletCount As Long
Dim lBulletNum As Long
Dim lExplCount As Long
Dim lExplNum As Long
Dim lRocketCount As Long
Dim lRocketNum As Long
Dim lActiveGuns As Long
Dim lLastShot As Long
Dim lGunCount As Long

'misc program variables
Dim bRunning As Boolean
Dim lEnemyCount As Long
Dim lEnemiesInRange As Long
Dim lFrameRate As Long

Private Sub Form_Load()
   Dim n As Long
   Dim fn As String
   Dim ret As Boolean

   CWHITE = RGB(255, 255, 255)
   CRED = RGB(255, 0, 0)
   CGREEN = RGB(0, 255, 0)
   CBLUE = RGB(0, 0, 255)
   CBLACK = RGB(0, 0, 0)

   Randomize GetTickCount
   bRunning = True

   'set up the main form
   Form1.Caption = "Seed of Freedom"
   Form1.AutoRedraw = False
   Form1.BorderStyle = 1
   Form1.ClipControls = False
   Form1.KeyPreview = True
   Form1.ScaleMode = 3
   Form1.Width = 640 * Screen.TwipsPerPixelX
   Form1.Height = 480 * Screen.TwipsPerPixelY
   Form1.Show

   'create the PictureBox control for drawing
```

```
Set Picture1 = Controls.Add("VB.PictureBox", "Picture1")
Picture1.AutoRedraw = False
Picture1.BorderStyle = 1
Picture1.ClipControls = False
Picture1.ScaleMode = 3
Picture1.BackColor = CBLACK
Picture1.Left = 0
Picture1.Top = 0
Picture1.Width = Form1.ScaleWidth
Picture1.Height = Form1.ScaleHeight
Picture1.Visible = True

'create DirectInput object
Set dinput = New clsDirectInput8
dinput.Startup Form1.hWnd

'create Keyboard object
Set Keyboard = New clsDIKeyboard8
Keyboard.Startup dinput, Form1.hWnd

'create Joystick object
Set Joystick = New clsDIJoystick8
If Not Joystick.Startup(dinput, Form1.hWnd) Then
   Debug.Print "Joystick not detected"
End If

'initialize DirectSound
Set dsound = New clsDirectSound8
dsound.Startup Form1.hWnd

'load fire sound
Set dswFire = New clsDSWave8
dswFire.LoadSound dsound, App.Path & "\sound\fire.wav"
dswFire.Interrupt = True

'load boom sound
Set dswBoom = New clsDSWave8
dswBoom.LoadSound dsound, App.Path & "\sound\boom.wav"
dswBoom.Interrupt = False

'load rocket sound
Set dswRocket = New clsDSWave8
dswRocket.LoadSound dsound, App.Path & "\sound\rocket.wav"
dswRocket.Interrupt = True

'initialize DirectMusic
```

```
Set dmusic = New clsDirectMusic8
If Not dmusic.Startup(Form1.hWnd) Then
   MsgBox "Error initializing DirectMusic"
   Shutdown
End If

'load music sequence
Set dmsSong = New clsDMSequence8
dmusic.Volume = -1500
PlayMissionSong

'create the double buffer
ret = cbBackBuf.Create(Picture1.hdc, Picture1.ScaleWidth, _
   Picture1.ScaleHeight)
If ret = False Or cbBackBuf.hdc = 0 Then
   MsgBox "Error creating double buffer"
   Shutdown
End If

'create the scroller object
Set Scroller = New clsVScroller

'load some stuff temporarily needed
LoadNewMission

'load the enemy planes
For n = 0 To MAXENEMIES
   Set csEnemy(n) = New clsSprite
   csEnemy(n).Active = True
   csEnemy(n).Transparent = True
   ret = csEnemy(n).LoadFrames(App.Path & "\bmp\ships.bmp", _
      1, 1, 32, 32, 1)
   If ret = False Or csEnemy(n).hdc = 0 Then
      MsgBox "Error loading enemy plane #" & n
      Shutdown
   End If

   csEnemy(n).GlobalX = Random(Scroller.MaxTileCols * _
      Scroller.TileWidth) - csEnemy(n).Width
   csEnemy(n).GlobalY = Random(Scroller.MaxTileRows * _
      Scroller.TileHeight) - csEnemy(n).Height
   csEnemy(n).SpeedX = Random(3) - 1
   csEnemy(n).SpeedY = 1
Next n
If n < MAXENEMIES Then
   MsgBox "Error loading enemy planes (count = " & n & ")"
```

```
    Shutdown
  End If

  'load the ground guns
  For n = 0 To MAXGUNS
    Set csGuns(n) = New clsSprite
    csGuns(n).Transparent = True
    csGuns(n).Active = True
    ret = csGuns(n).LoadFrames(App.Path & "\bmp\enemies.bmp", _
      1, 34, 32, 32, 1)
    If ret = False Or csGuns(n).hdc = 0 Then
      MsgBox "Error loading ships.bmp"
      Shutdown
    End If

    csGuns(n).GlobalX = Random(Scroller.MaxTileCols * _
      Scroller.TileWidth) - csGuns(n).Width
    csGuns(n).GlobalY = Random(Scroller.MaxTileRows * _
      Scroller.TileHeight) - csGuns(n).Height
    csGuns(n).SpeedX = 0
    csGuns(n).SpeedY = 0
  Next n
  If n < MAXGUNS Then
    MsgBox "Error loading enemy guns (count = " & n & ")"
    Shutdown
  End If

  'load the player bullets
  For lBulletNum = 0 To MAXBULLETS
    Set csPlayerBullets(lBulletNum) = New clsSprite
    csPlayerBullets(lBulletNum).Transparent = True
    ret = csPlayerBullets(lBulletNum).LoadFrames( _
      App.Path & "\bmp\bullets.bmp", 0, 0, 32, 8, 1)
    If ret = False Or _
      csPlayerBullets(lBulletNum).hdc = 0 Then
      MsgBox "Error loading bullets.bmp"
      Shutdown
    End If
  Next lBulletNum
  If lBulletNum < MAXBULLETS Then
    MsgBox "Error loading player bullets (count = " & _
      lBulletNum & ")"
    Shutdown
  End If

  'load the enemy bullets
```

```
For lBulletNum = 0 To MAXBULLETS
   Set csEnemyBullets(lBulletNum) = New clsSprite
   csEnemyBullets(lBulletNum).Transparent = True
   ret = csEnemyBullets(lBulletNum).LoadFrames( _
      App.Path & "\bmp\bullets.bmp", 0, 9, 32, 8, 1)
   If ret = False Or _
      csEnemyBullets(lBulletNum).hdc = 0 Then
      MsgBox "Error loading bullets.bmp"
      Shutdown
   End If
Next lBulletNum
If lBulletNum < MAXBULLETS Then
   MsgBox "Error loading enemy bullets (count = " & _
      lBulletNum & ")"
   Shutdown
End If

'load the enemy rockets
For lBulletNum = 0 To MAXROCKETS
   Set csEnemyRockets(lBulletNum) = New clsSprite
   csEnemyRockets(lBulletNum).Transparent = True
   ret = csEnemyRockets(lBulletNum).LoadFrames( _
      App.Path & "\bmp\bullets.bmp", 0, 18, 8, 16, 1)
   If ret = False Or _
      csEnemyRockets(lBulletNum).hdc = 0 Then
      MsgBox "Error loading bullets.bmp"
      Shutdown
   End If
Next lBulletNum
If lBulletNum < MAXROCKETS Then
   MsgBox "Error loading enemy rockets (count = " & _
      lBulletNum & ")"
   Shutdown
End If

'load the explosions
For n = 0 To MAXEXPLODES
   Set csExplosions(n) = New clsSprite
   csExplosions(n).Transparent = True
   csExplosions(n).Active = False
   ret = csExplosions(n).LoadFrames( _
      App.Path & "\bmp\explosions.bmp", 0, 0, 17, 16, 9)
   If ret = False Or csExplosions(n).hdc = 0 Then
      MsgBox "Error loading explosions.bmp"
      Shutdown
   End If
```

```
    Next n
    If n < MAXEXPLODES Then
        MsgBox "Error loading explosions (count = " & n & ")"
        Shutdown
    End If

    lMissionNumber = 0
    lCurrentMission = lMissionNumber
    lPlayerScore = 0
    GameStatus = SHOWTITLE

    GameLoop
End Sub

Private Sub Form_Paint()
    cbBackBuf.Draw 0, 0, Picture1.hdc
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, _
    UnloadMode As Integer)
    Shutdown
End Sub

Private Sub Keyboard_KeyDown(ByVal lKey As Long)
    Select Case lKey
        Case KEY_LEFT
            MovePlane KEY_LEFT

        Case KEY_RIGHT
            MovePlane KEY_RIGHT

        Case KEY_UP
            MovePlane KEY_UP

        Case KEY_DOWN
            MovePlane KEY_DOWN

        Case KEY_LCTRL
            'fire a shot
            If GameStatus = RUNNING Then PlayerFireShot

        Case KEY_ESC
            Shutdown

        Case KEY_PGDN
            GameStatus = NEXTMISSION
```

```
    Case KEY_SPACE
      Select Case GameStatus
        Case SHOWTITLE
          'display mission screen
          bMissionLoaded = False
          GameStatus = NEXTMISSION
        Case NEXTMISSION
          'start next mission
          If lMissionNumber = 10 Then
            Shutdown
          Else
            GameStatus = LOADING
          End If
        Case ENDGAME
          'end game
          Shutdown
      End Select
  End Select
End Sub

Private Sub Joystick_ButtonDown(ByVal lButton As Long)
  'fire a shot
  If GameStatus = RUNNING Then PlayerFireShot
End Sub

Private Sub Joystick_DPAD(ByVal lButton As Long)
  Static lLastTime As Long

  If lLastTime + 50 < GetTickCount Then
    lLastTime = GetTickCount
    Select Case lButton
      Case -1 'release
        MovePlane ALLSTOP

      Case 0 'up
        MovePlane KEY_UP

      Case 1 'up/right
        MovePlane KEY_UP
        MovePlane KEY_RIGHT

      Case 2 'right
        MovePlane KEY_RIGHT

      Case 3 'down/right
```

```
            MovePlane KEY_RIGHT
            MovePlane KEY_DOWN

        Case 4 'down
            MovePlane KEY_DOWN

        Case 5 'down/left
            MovePlane KEY_DOWN
            MovePlane KEY_LEFT

        Case 6 'left
            MovePlane KEY_LEFT

        Case 7 'up/left
            MovePlane KEY_LEFT
            MovePlane KEY_UP
    End Select
  End If
End Sub

Private Sub Scroller_Complete()
  'if game loop running, go to next mission
  If GameStatus = RUNNING Then
    GameStatus = NEXTMISSION
    bMissionLoaded = False
  End If
End Sub

Private Sub GameLoop()
  Static lStartTime As Long
  Static lCounter As Long
  Static lNewTime As Long
  Dim fn As String

  'store starting time
  lStartTime = GetTickCount

  'game loop
  Do While bRunning
    lCounter = GetTickCount() - lStartTime
    'if time for next frame...
    If lCounter > lNewTime Then
      'check the game status
      Select Case GameStatus
        Case SHOWTITLE
          If Not bMissionLoaded Then
```

```
        lMissionNumber = 0
        lCurrentMission = lMissionNumber
        LoadNewMission
        bMissionLoaded = True
    End If
    Scroller.Scroll 1
    Scroller.DrawTiles 13, 10, cbBackBuf.hdc
    csMission.X = 160
    csMission.Y = 120
    csMission.Draw cbBackBuf.hdc
    cbBackBuf.DrawText Picture1.ScaleWidth _
        / 2 - 80, Picture1.ScaleHeight - 40, _
        "Press SPACEBAR To Continue", CBLUE
    DoEvents

Case NEXTMISSION
    If Not bMissionLoaded Then
        lMissionNumber = lMissionNumber + 1
        lCurrentMission = lMissionNumber
        LoadNewMission
        ResetEnemies
        bMissionLoaded = True
        PlayMissionSong
    End If
    Scroller.Scroll 1
    Scroller.DrawTiles 13, 10, cbBackBuf.hdc
    csMission.X = 160
    csMission.Y = 120
    csMission.Draw cbBackBuf.hdc
    cbBackBuf.DrawText Picture1.ScaleWidth _
        / 2 - 80, Picture1.ScaleHeight - 40, _
        "Press SPACEBAR To Continue", CBLUE
    DoEvents

Case LOADING
    Set csPlane = Nothing
    Set csPlane = New clsSprite
    csPlane.Transparent = True
    fn = App.Path & "\bmp\planes.bmp"
    If Not csPlane.LoadFrames(fn, 0, (lMissionNumber - 1) _
        * 32, 32, 32, 3) Then
        MsgBox "Error loading " & fn
        Shutdown
    End If

    'move fighter to starting position
```

```
            csPlane.CurrentFrame = 1
            csPlane.X = 320 - csPlane.Width / 2
            csPlane.Y = Picture1.ScaleHeight - _
               csPlane.Height * 2

            'LoadNewMission
            bMissionLoaded = False
            GameStatus = RUNNING
            DoEvents

         Case RUNNING
            Scroller.Scroll 1
            Scroller.DrawTiles 13, 10, cbBackBuf.hdc
            GameUpdate lCounter
            lNewTime = lCounter + 1000 / MAXFRAMERATE
            DoEvents

      End Select

      'blit double buffer to the screen
      cbBackBuf.Draw 0, 0, Picture1.hdc

      'check for user input
      Keyboard.Check_Keyboard
      Joystick.Check_Joystick
   End If
   DoEvents
  Loop
End Sub

Private Sub GameUpdate(ByVal MS As Long)
  Static lCounter As Long
  Static lTimer As Long
  Static lStart As Long
  Static lScroll As Long
  Dim lLeft As Long
  Dim lValue As Long
  Dim n As Long

  'begin screen update timing
  lStart = GetTickCount

  'process game objects
  MoveEnemies
  MoveGuns
  MoveBullets
```

```
    MoveRockets
    MovePlayer
    DrawExplosions

    'display player damage level
    'lLeft = Picture1.ScaleWidth - 70
    'lValue = lPlayerHealth - lPlayerDamage
    'If lValue > 0 Then
      'For n = 1 To lValue * 5
        'cbBackBuf.DrawLine lLeft + n, 10, _
          lLeft + n, 20, CRED
      'Next n
      'cbBackBuf.DrawRect lLeft, 10, lLeft + 50, _
        '20, CWHITE
  ' End If

    'display score and timer
    cbBackBuf.DrawText 2, 2, "SCORE " & lPlayerScore, CWHITE
    cbBackBuf.DrawText Picture1.ScaleWidth / 2 - 40, 10, _
      "TIME " & Scroller.MajorScroll, CWHITE

    'display game info
    'cbBackBuf.DrawText 2, 15, "Explosions " & _
      'lExplCount, CBLACK
    'cbBackBuf.DrawText 2, 28, "Shots " & lBulletCount, CBLACK
    'cbBackBuf.DrawText 2, 41, "Enemies " & lEnemyCount & _
      '" in range " & lEnemiesInRange, CBLACK
    'cbBackBuf.DrawText 2, 54, "Guns " & lGunCount & _
      '" in range " & lActiveGuns, CBLACK
    'cbBackBuf.DrawText 2, 67, "FPS " & lFrameRate, CBLACK
    'cbBackBuf.DrawText 2, 80, "Player " & csPlane.X & _
      "," & csPlane.Y, CBLACK

    'end screen update timing
    lStart = GetTickCount - lStart

    'count the frames per second
    If MS > lTimer + 1000 Then
      lTimer = MS
      lFrameRate = lCounter
      lCounter = 0
    Else
      lCounter = lCounter + 1
    End If
  End Sub
```

```
Private Sub PlayMissionSong()
   Dim fn As String

   'stop currently playing music
   dmsSong.StopMusic

   'load MIDI file
   fn = App.Path & "\sound\mission" & lMissionNumber & ".mid"
   If Not dmsSong.LoadMusic(dmusic, fn) Then
      MsgBox "Error loading MIDI file " & fn
      Shutdown
   End If

   'start song playing in a loop
   dmsSong.Looping = True
   dmsSong.PlayMusic
End Sub

Private Sub ResetEnemies()
   Dim n As Long

   'reset enemy planes
   For n = 0 To MAXENEMIES
      csEnemy(n).Active = True
      csEnemy(n).GlobalX = Random(Scroller.MaxTileCols * _
         Scroller.TileWidth) - csEnemy(n).Width
      csEnemy(n).GlobalY = Random(Scroller.MaxTileRows * _
         Scroller.TileHeight) - csEnemy(n).Height
   Next n

   'reset enemy bullets
   For n = 0 To MAXBULLETS
      With csEnemyBullets(n)
         .Active = False
         .X = 0
         .Y = 0
      End With
   Next n

   'reset enemy guns
   For n = 0 To MAXGUNS
      csGuns(n).Active = True
      csGuns(n).GlobalX = Random(Scroller.MaxTileCols * _
         Scroller.TileWidth) - csEnemy(n).Width
      csGuns(n).GlobalY = Random(Scroller.MaxTileRows * _
         Scroller.TileHeight) - csEnemy(n).Height
```

```
    Next n

    'reset enemy rockets
    For n = 0 To MAXROCKETS
        With csEnemyRockets(n)
            .Active = False
            .X = 0
            .Y = 0
        End With
    Next n
End Sub

Private Sub LoadNewMission()
    Dim fn As String

    're-create the mission graphic
    Set csMission = Nothing
    Set csMission = New clsSprite

    'load the mission graphic
    csMission.Transparent = True
    fn = App.Path & "\bmp\mission" & lMissionNumber & ".bmp"
    If Not csMission.LoadFrames(fn, 0, 0, 320, 200, 1) Then
        MsgBox "Error loading " & fn
        Shutdown
    End If

    'load tiles for current map
    fn = App.Path & "\bmp\plaintiles.bmp"
    If Not Scroller.LoadTiles(fn, 0, 0, 48, 48, 10) Then
        MsgBox "Error loading " & fn
        Shutdown
    End If

    'set left edge of map
    Scroller.StartX = 2

    'load the scrolling terrain
    CreateRandomLevel App.Path & "\level.dat", 13, 50 + _
        lMissionNumber * 10
    LoadLevel App.Path & "\level.dat"

    'initialize plane stats
    Select Case lMissionNumber
        Case 1 'P-40
            lPlayerHealth = 200
```

```
    lPlayerDamage = 0
    lPlayerAttack = 4
    lPlayerSpeed = 1

Case 2 'P-61
    lPlayerHealth = 300
    lPlayerDamage = 0
    lPlayerAttack = 4
    lPlayerSpeed = 1

Case 3 'P-63
    lPlayerHealth = 300
    lPlayerDamage = 0
    lPlayerAttack = 4
    lPlayerSpeed = 1

Case 4 'Corsair
    lPlayerHealth = 400
    lPlayerDamage = 0
    lPlayerAttack = 4
    lPlayerSpeed = 2

Case 5 'P-36
    lPlayerHealth = 400
    lPlayerDamage = 0
    lPlayerAttack = 5
    lPlayerSpeed = 2

Case 6 'P-39
    lPlayerHealth = 500
    lPlayerDamage = 0
    lPlayerAttack = 6
    lPlayerSpeed = 2

Case 7 'P-38
    lPlayerHealth = 700
    lPlayerDamage = 0
    lPlayerAttack = 7
    lPlayerSpeed = 3

Case 8 'P-51
    lPlayerHealth = 700
    lPlayerDamage = 0
    lPlayerAttack = 8
    lPlayerSpeed = 3
```

```
      Case 9 'P-47
        lPlayerHealth = 900
        lPlayerDamage = 0
        lPlayerAttack = 9
        lPlayerSpeed = 3
    End Select
End Sub

Private Sub DrawExplosions()
    Dim n As Long
    Dim lNum As Long

    lExplCount = 0
    For lNum = 0 To MAXEXPLODES
      With csExplosions(lNum)
        'make sure explosion is alive
        If .Active Then
          'play explosion sound
          dswBoom.PlaySound
          'increment explosion counter
          lExplCount = lExplCount + 1
          'draw explosion sprite five times
          For n = 0 To 5
            .X = .X + Random(3) - 1
            .Y = .Y + Random(3) - 1
            .Draw cbBackBuf.hdc
            .CurrentFrame = Random(10)
          Next n
          'draw multiple times for effect
          .MoveRate = .MoveRate - 1
          If .MoveRate < 1 Then .Active = False
        End If
      End With
    Next lNum
End Sub

Private Sub MovePlane(ByVal lDir As Long)
    With csPlane
      Select Case lDir
        Case KEY_LEFT
          .SpeedX = -lPlayerSpeed
        Case KEY_RIGHT
          .SpeedX = lPlayerSpeed
        Case KEY_UP
          .SpeedY = -lPlayerSpeed
        Case KEY_DOWN
```

```
            .SpeedY = lPlayerSpeed
        Case Else
            .SpeedX = 0
            .SpeedY = 0
    End Select
  End With
End Sub

Private Sub MovePlayer()
  With csPlane
    'set the current frame
    .CurrentFrame = .SpeedX + 1

    'update X position
    .X = .X + .SpeedX
    If .X < 1 Then
      .X = 1
    ElseIf .X > Picture1.ScaleWidth - .Width - 1 Then
      .X = Picture1.ScaleWidth - .Width - 1
    End If
    .SpeedX = 0

    'update Y position
    .Y = .Y + .SpeedY
    If .Y < 1 Then
      .Y = 1
    ElseIf .Y > Picture1.ScaleHeight - .Height - 1 Then
      .Y = Picture1.ScaleHeight - .Height - 1
    End If
    .SpeedY = 0

    'draw sprite
    .Draw cbBackBuf.hdc
  End With
End Sub

Private Sub PlayerFireShot()
  'see if a sprite is available
  For lBulletNum = 0 To MAXBULLETS
    If Not csPlayerBullets(lBulletNum).Active Then
      Exit For
    End If
  Next lBulletNum

  'delay 1/10 second between shots
  If lLastShot + 100 < GetTickCount Then
```

```
      lLastShot = GetTickCount
      'limit number of shots
      If lBulletNum <= lPlayerAttack Then
         'play fire sound
         dswFire.PlaySound
         'set up projectile
         With csPlayerBullets(lBulletNum)
            .Active = True
            .X = csPlane.X
            .Y = csPlane.Y
            .SpeedX = 0
            .SpeedY = -3
            .MoveRate = GetTickCount
         End With
      End If
   End If
End Sub

Private Sub EnemyFireShot(ByVal lNum As Long)
   Static lLastShot As Long
   Dim num As Long

   'delay between shots
   If GetTickCount > lLastShot + 400 Then
      lLastShot = GetTickCount

      'check for available sprites
      For lBulletNum = 0 To MAXBULLETS
         If Not csEnemyBullets(lBulletNum).Active Then
            Exit For
         End If
      Next lBulletNum

      'limit number of shots
      If lBulletNum <= MAXBULLETS Then
         'play shot sound
         dswFire.PlaySound
         'set up projectile
         With csEnemyBullets(lBulletNum)
            .Active = True
            .X = csEnemy(lNum).X
            .Y = csEnemy(lNum).Y + csEnemy(lNum).Height
            .SpeedX = 0
            .SpeedY = 3
            .MoveRate = GetTickCount
         End With
```

```
      End If
   End If
End Sub

Private Sub GunFireRocket(ByVal lNum As Long)
   Static lLastShot As Long
   Dim num As Long

   'delay between shots
   If GetTickCount > lLastShot + 400 Then
      lLastShot = GetTickCount

      'check for available sprites
      For lRocketNum = 0 To MAXBULLETS
         If Not csEnemyRockets(lRocketNum).Active Then
            Exit For
         End If
      Next lRocketNum

      'limit number of shots
      If lRocketNum <= MAXBULLETS Then
         'play shot sound
         dswRocket.PlaySound
         'set up the rocket
         With csEnemyRockets(lRocketNum)
            .Active = True
            .X = csGuns(lNum).X + 14
            .Y = csGuns(lNum).Y + 10
            .SpeedX = 0
            .SpeedY = 3
            .MoveRate = GetTickCount
         End With
      End If
   End If
End Sub

Private Sub StartExplosion(ByRef spr As clsSprite)
   'check for available sprites
   For lExplNum = 0 To MAXEXPLODES
      If Not csExplosions(lExplNum).Active Then
         Exit For
      End If
   Next lExplNum

   'limit number of explosions
   If lExplNum <= MAXEXPLODES Then
```

```
      'set up the explosion
      With csExplosions(lExplNum)
        .Active = True
        .MoveRate = 20
        .X = spr.X + Random(spr.Width - 8)
        .Y = spr.Y + Random(spr.Height - 8)
      End With
    End If
End Sub

Private Sub MoveRockets()
  For lRocketNum = 0 To MAXROCKETS
    With csEnemyRockets(lRocketNum)
      'make sure rocket is alive
      If .Active Then
        'update rocket position
        .X = .X + .SpeedX
        If .X < 1 Or .X > Picture1.ScaleWidth Then
          .Active = False
        End If
        .Y = .Y + .SpeedY
        If .Y < 1 Or .Y > Picture1.ScaleHeight Then
          .Active = False
        End If

        'draw the rocket
        .Draw cbBackBuf.hdc

        'check for collision with player
        If .Collided(csPlane) Then
          .Active = False
          lPlayerDamage = lPlayerDamage + 1
          StartExplosion csPlane
          Debug.Print "Rocket " & lBulletNum & _
            " hit PLAYER at " & .X & "," & .Y

          'see if plane was destroyed (restart mission)
          If lPlayerHealth - lPlayerDamage < 0 Then
            bMissionLoaded = False
            lMissionNumber = lCurrentMission - 1
            GameStatus = NEXTMISSION
            lPlayerDamage = 0
          End If
        End If
      End If
    End With
```

```
    Next lRocketNum
End Sub

Private Sub MoveBullets()
    Dim n As Long

    lBulletCount = 0
    For lBulletNum = 0 To MAXBULLETS
        'move the player's shots
        With csPlayerBullets(lBulletNum)
            'make sure bullet is alive
            If .Active Then
                'update bullet position
                lBulletCount = lBulletCount + 1
                .X = .X + .SpeedX
                If .X < 1 Or .X > Picture1.ScaleWidth Then
                    .Active = False
                End If
                .Y = .Y + .SpeedY
                If .Y < 1 Or .Y > Picture1.ScaleHeight Then
                    .Active = False
                End If

                'draw the bullet sprite
                .Draw cbBackBuf.hdc

                'check for collision with enemy planes
                For n = 0 To MAXENEMIES
                    If csEnemy(n).Active Then
                        If .Collided(csEnemy(n)) Then
                            .Active = False
                            csEnemy(n).Active = False
                            StartExplosion csEnemy(n)
                            lPlayerScore = lPlayerScore + 50
                            Debug.Print "Bullet " & lBulletNum & _
                                " hit ENEMY at " & .X & "," & .Y
                            Exit For
                        End If
                    End If
                Next n

                'check for collision with enemy guns
                For n = 0 To MAXGUNS
                    If csGuns(n).Active And csGuns(n).X <> 0 And _
                        csGuns(n).Y <> 0 Then
                        If .Collided(csGuns(n)) Then
```

```
            .Active = False
            csGuns(n).Active = False
            StartExplosion csGuns(n)
            lPlayerScore = lPlayerScore + 75
            Debug.Print "Bullet " & lBulletNum & _
               " hit GUN at " & .X & "," & .Y
            Exit For
         End If
      End If
   Next n
 End If
End With

'move enemy bullets
With csEnemyBullets(lBulletNum)
  'make sure bullet is alive
  If .Active Then
    'update bullet position
    .X = .X + .SpeedX
    If .X < 1 Or .X > Picture1.ScaleWidth Then
      .Active = False
    End If
    .Y = .Y + .SpeedY
    If .Y < 1 Or .Y > Picture1.ScaleHeight Then
      .Active = False
    End If

    'draw enemy bullet sprite
    .Draw cbBackBuf.hdc

    'check for collision with player
    If .Collided(csPlane) Then
      .Active = False
      lPlayerDamage = lPlayerDamage + 1
      StartExplosion csPlane
      Debug.Print "Bullet " & lBulletNum & _
        " hit PLAYER at " & .X & "," & .Y

      'see if plane was destroyed (restart mission)
      If lPlayerHealth - lPlayerDamage < 0 Then
        bMissionLoaded = False
        lMissionNumber = lCurrentMission - 1
        GameStatus = NEXTMISSION
        lPlayerDamage = 0
      End If
    End If
```

```
        End If
      End With
    Next lBulletNum
End Sub

Private Sub MoveEnemies()
    Dim n As Long
    lEnemyCount = 0
    lEnemiesInRange = 0
    For n = 0 To MAXENEMIES
      With csEnemy(n)
        'make sure enemy plane is alive
        If .Active Then
          lEnemyCount = lEnemyCount + 2

          'update enemy plane position
          .GlobalX = .GlobalX + .SpeedX
          .GlobalY = .GlobalY + .SpeedY

          'rebound off screen edges
          If .GlobalX < 1 Then
            .GlobalX = 1
            .SpeedX = 1
          ElseIf .GlobalX > Picture1.ScaleWidth - 1 Then
            .GlobalX = Picture1.ScaleWidth - 1
            .SpeedX = -1
          End If

          'warp enemy plane up to top of map
          If .GlobalY > Scroller.MaxTileRows * _
            Scroller.TileHeight Then
            .GlobalY = 1
          End If

          'draw the enemy plane if it is in range
          If .GlobalY > (Scroller.MajorScroll - 1) * _
            Scroller.TileHeight Then
            lEnemiesInRange = lEnemiesInRange + 1
            .X = .GlobalX
            .Y = .GlobalY - Scroller.MajorScroll * _
              Scroller.TileHeight + Scroller.MinorScroll
            .Draw cbBackBuf.hdc

            'fire at player if in range
            If .Y + .Height < csPlane.Y Then
              If .X > csPlane.X - 30 And .X < _
```

```
                csPlane.X + csPlane.Width + 30 Then
                EnemyFireShot n
              End If
            End If
          End If
        End If
      End With
    Next n
End Sub

Private Sub MoveGuns()
    Dim n As Long
    lGunCount = 0
    lActiveGuns = 0
    For n = 0 To MAXGUNS
      With csGuns(n)
        'make sure gun is active
        If .Active Then
          'update gun position
          lGunCount = lGunCount + 2
          .GlobalY = .GlobalY + .SpeedY
          'wrap to top of map if it reaches the bottom
          If .GlobalY > Scroller.MaxTileRows * _
            Scroller.TileHeight Then
            .GlobalY = 1
          End If
          'draw the gun if it is in range
          If .GlobalY > (Scroller.MajorScroll - 1) * _
            Scroller.TileHeight Then
            lActiveGuns = lActiveGuns + 1
            .X = .GlobalX
            .Y = .GlobalY - Scroller.MajorScroll * _
              Scroller.TileHeight + Scroller.MinorScroll
            .Draw cbBackBuf.hdc

            'look for player
            If .Y + .Height < csPlane.Y Then
              If .X > csPlane.X - 30 And .X < _
                csPlane.X + csPlane.Width + 30 Then
                GunFireRocket n
              End If
            End If
          End If
        End If
      End With
    Next n
```

```
End Sub

Public Function Random(ByVal num&) As Long
   Random = CLng(num * Rnd)
End Function

Public Sub CreateRandomLevel(ByVal sFilename As String, _
   ByVal lMaxCols As Long, ByVal lMaxRows As Long)
   Dim lTileArray() As Long
   Dim lFileNum As Long
   Dim rows As Long
   Dim cols As Long
   Dim lTile As Long
   Dim lWidth As Long
   Dim lHeight As Long
   Dim X As Long
   Dim Y As Long

   On Error GoTo error1
   'redimension tile array
   ReDim lTileArray(0 To lMaxCols + 5, 0 To lMaxRows + 5)

   'create random tile values
   For rows = 1 To lMaxRows
      For cols = 1 To lMaxCols
         lTile = Random(4)
         If lTile = 1 Then
            lTile = 9
         Else
            lTile = 0
         End If
         lTileArray(cols, rows) = lTile
      Next cols
   Next rows

   'place random tiles in larger virtual map
   For lTile = 1 To 20
      lWidth = Random(3) + 1
      lHeight = Random(3) + 1
      X = Random(lMaxCols)
      Y = Random(lMaxRows)
      lTileArray(X, Y) = 7
      lTileArray(X, Y + lHeight) = 1
      lTileArray(X + lWidth, Y) = 5
      lTileArray(X + lWidth, Y + lHeight) = 3
      If lWidth = 3 Then
```

```
            lTileArray(X + 2, Y) = 6
            lTileArray(X + 2, Y + lHeight) = 2
        End If
        If lWidth >= 2 Then
            lTileArray(X + 1, Y) = 6
            lTileArray(X + 1, Y + lHeight) = 2
        End If
        If lHeight = 3 Then
            lTileArray(X, Y + 2) = 8
            lTileArray(X + lWidth, Y + 2) = 4
        End If
        If lHeight >= 2 Then
            lTileArray(X, Y + 1) = 8
            lTileArray(X + lWidth, Y + 1) = 4
        End If
    Next lTile

    'write the tile values out to a file
    lFileNum = FreeFile()
    Open sFilename For Output As #lFileNum
    Write #lFileNum, lMaxCols; lMaxRows
    For rows = 1 To lMaxRows
        For cols = 1 To lMaxCols
            Write #lFileNum, lTileArray(cols, rows);
        Next cols
    Next rows

    Close #lFileNum
error1:
End Sub

Public Function LoadLevel( _
    ByVal sFilename As String) As Boolean
    Dim numcols As Long
    Dim numrows As Long
    Dim rows As Long
    Dim cols As Long
    Dim lFileNum As Long
    Dim lTileValue As Long
    Dim sText As String
    Dim lLine As Long

    On Error GoTo error1
    LoadLevel = False

    'open level file and load tiles
```

```
    lFileNum = FreeFile()
    Open sFilename For Input As #lFileNum
    Input #lFileNum, numcols, numrows
    Scroller.InitTileMap numcols, numrows
    For rows = 0 To numrows - 1
        'load map tile values
        For cols = 0 To numcols - 1
            Input #lFileNum, lTileValue
            Scroller.SetTile cols, rows, lTileValue
        Next cols
    Next rows

    Close #lFileNum
    LoadLevel = True
error1:
End Function

Private Sub Shutdown()
    Dim n As Long

    'stop game loop
    bRunning = False

    'delete objects
    Set Scroller = Nothing
    Set cbBackBuf = Nothing
    Set csPlane = Nothing
    Set csMission = Nothing

    'delete bullets
    For n = 0 To MAXBULLETS
        Set csPlayerBullets(n) = Nothing
        Set csEnemyBullets(n) = Nothing
    Next n

    'delete rockets
    For n = 0 To MAXROCKETS
        Set csEnemyRockets(n) = Nothing
    Next n

    'delete enemy planes
    For n = 0 To MAXENEMIES
        Set csEnemy(n) = Nothing
    Next n

    'delete DirectInput objects
```

```
    Joystick.Shutdown
    Set Joystick = Nothing
    Keyboard.Shutdown
    Set Keyboard = Nothing
    Set dinput = Nothing

    'delete DirectSound objects
    Set dsound = Nothing
    Set dswFire = Nothing
    Set dswBoom = Nothing
    Set dswRocket = Nothing

    'delete DirectMusic objects
    dmsSong.StopMusic
    Set dmsSong = Nothing
    Set dmusic = Nothing

    'end program
    Form1.Hide
    End
End Sub
```