SKEET SHOOTERS

VIDEO GAMING SOFTWARE
XBOX 360 VIDEO GAME CONSOLE

Josh Yanai

CEN 4935 – Senior Software Engineering Project

Janusz Zalewski, Ph.D.

Florida Gulf Coast University

Spring 2011

Table of Contents

1. Introduction

This project is a continuation of a video game project called Skeet Shooters that was developed in the Computer Graphics (CAP 4730) course during the spring 2010 semester. Skeet Shooters is a gaming program, written in C# and using Microsoft's XNA framework, which runs on the Windows (XP or later) operating system. The game is to be further developed and converted to run on the XBox 360 video game console (Figure 1.1). The following hardware and software components necessary for completing this project are listed below.

1.1. Hardware

- Windows PC (XP or later) for game development and testing.
- XBox 360 video game console (Figure 1.1).

  The XBox 360 video game console "is the second video game console produced by Microsoft, and the successor to the XBox. The XBox 360 competes with Sony's PlayStation 3 and Nintendo's Wii as part of the seventh generation of video game consoles."[4]

Figure 1.1 Xbox 360 video game console [5]

- Mad Catz wireless gaming adapter (Figure 1.2).

  The Mad Catz wireless gaming adapter provides a wireless broadband Internet connection for the Xbox 360. The adapter connects to the Xbox 360 via Ethernet port and is compatible with 802.11 b/g/n wireless networks.

Figure 1.2 Mad Catz wireless gaming adapter [6]

1.2. Software

Microsoft Visual Studio 2008

"Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It can be used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight."[2]

Microsoft XNA framework

"Microsoft XNA is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development and management. XNA attempts to free game developers from writing "repetitive boilerplate code" and to bring different aspects of game production into a single system."[3]

2. Problem Description

2.1. Project Objective and Features

The objective of this project is to port the Skeet Shooters gaming software to run on the XBox 360 gaming system. Other additions to the game include:

- Adding smooth and visually appealing screen transitions.
- Changing the main menu screen (Figure 2.2)
- Adding an options menu screen (Figure 2.3)
- Adding difficulty level selection (beginner, intermediate, expert) (Figure 2.3)
- Adding an interactive game tutorial (Figure 2.10)
- Adding a sound menu screen (Figure 2.4)
- Adding a music volume menu screen (Figure 2.5)
- Adding a sound effects volume menu screen (Figure 2.6)
- Adding a pause menu screen (Figure 2.7)
- Adding a more game sounds and music
- Making minor graphical adjustments/changes
- Making minor game play adjustments/changes

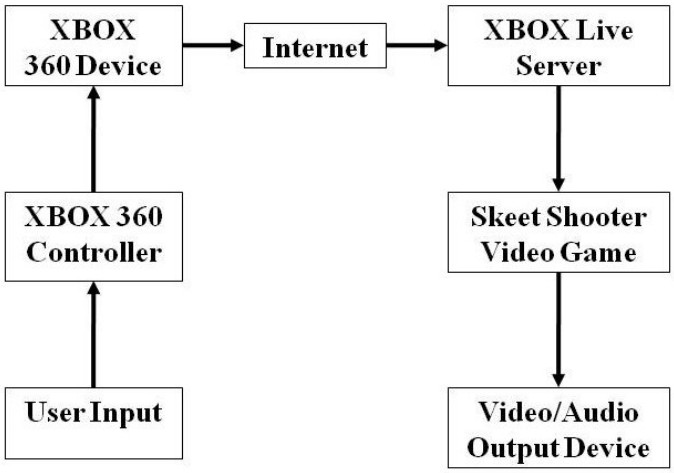


Figure 2.1 Context Diagram

2.2. Functional Requirements

2.2.1. The video game shall run on the Xbox 360 video game console.

2.2.2. The video game shall have multiple menu screens.

2.2.3. Each menu screen shall have multiple menu entries.

2.2.3.1. The top-most menu entry of a menu screen shall be highlighted initially.

2.2.3.2. Pressing UP on the Directional Pad (Figure 2.11) shall highlight the menu entry above the currently highlight entry.

2.2.3.3. If the currently highlighted menu entry is the top-most menu entry, then the bottom-most menu entry shall be highlighted.

2.2.3.4. Pressing DOWN on the Directional Pad (Figure 2.11) shall highlight the menu entry below the currently highlight entry.

2.2.3.5. If the currently highlighted menu entry is the bottom-most menu entry, then the top-most menu entry shall be highlighted.

2.2.3.6. Pressing the A button (Figure 2.11) shall select the currently highlighted menu entry.

2.2.4. Menu Screen: Main Menu Screen (Figure 2.2) The main menu screen shall be automatically displayed at the start of the game.

2.2.4.1. Selecting the “Start Game” menu entry shall display the character selection screen.

2.2.4.2. Selecting the “Options” menu entry shall display the options menu screen.

2.2.4.3. Selecting the “Exit” menu entry shall exit the game.

Figure 2.2 Main Menu Screen

2.2.5. Menu Screen: Options Menu Screen (Figure 2.3) The options menu screen shall be displayed when the “Options” menu entry is selected from the main menu screen.

2.2.5.1. Selecting the “Difficulty” menu entry shall change the game difficulty and display the current difficulty setting.

2.2.5.2. Selecting the “Tutorial” menu entry shall begin the interactive game tutorial.

2.2.5.3. Selecting the “Sound” menu entry shall display the sound menu screen.

2.2.5.4. Selecting the “Back” menu entry shall display the main menu screen.

Figure 2.3 Options Menu Screen

2.2.6. Menu Screen: Sound Menu Screen (Figure 2.4) The sound menu screen shall be displayed when the “Sound” menu entry is selected from the options menu screen.

2.2.6.1. Selecting the “Music Volume” menu entry shall display the music volume menu screen.

2.2.6.2. Selecting the “SFX Volume” menu entry shall display the sound effects volume menu screen.

2.2.6.3. Selecting the “Next Song” menu entry shall play the next song in the music queue.

2.2.6.4. Selecting the “Prev Song” menu entry shall play the previous song in the music queue.

2.2.6.5. Selecting the “Back” menu entry shall display the options menu screen.

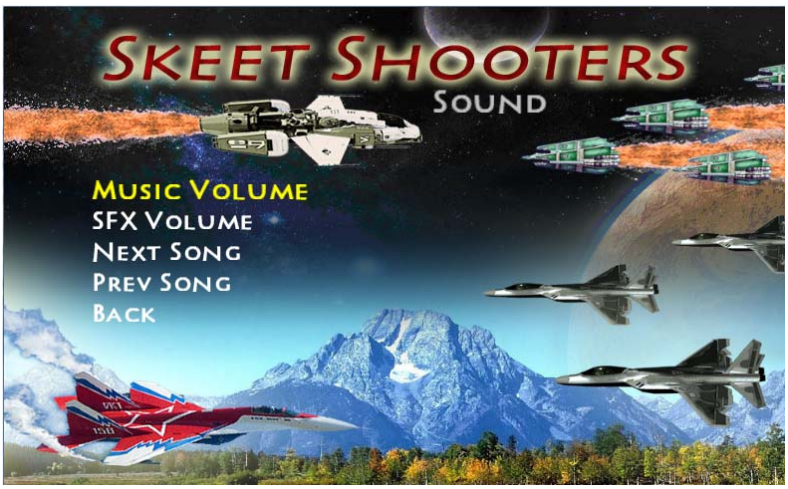


Figure 2.4 Sound Menu Screen

2.2.7. Menu Screen: Music Volume Menu Screen (Figure 2.5) The music volume menu screen shall be displayed when the "Music Volume" menu entry is selected from the sound menu screen.

2.2.7.1. Selecting the "Volume Up" menu entry shall increase the music volume by 10% and display the current music volume.

2.2.7.2. Maximum music volume shall be 100%.

2.2.7.3. Selecting the "Volume Down" menu entry shall decrease the music volume by 10% and display the current music volume.

2.2.7.4. Minimum music volume shall be 0%.

2.2.7.5. Selecting the "Back" menu entry shall display the sound menu screen.

Figure 2.5 Music Volume Menu Screen

2.2.8. Menu Screen: Sound Effects Volume Menu Screen (Figure 2.6) The sound effects volume menu screen shall be displayed when the “SFX Volume” menu entry is selected from the sound menu screen.

2.2.8.1. Selecting the “Volume Up” menu entry shall increase the sound effect volume by 10% and display the current sound effect volume.

2.2.8.2. Maximum sound effect volume shall be 100%

2.2.8.3. Selecting the “Volume Down” menu entry shall decrease the sound effect volume by 10% and display the current sound effect volume.

2.2.8.4. Minimum sound effect volume shall be 0%

2.2.8.5. Selecting the “Back” menu entry shall display the sound menu screen.

Figure 2.6 Sound Effects Volume Menu Screen

2.2.9. Menu Screen: Pause Menu Screen (Figure 2.7) The pause menu screen shall be displayed when the START button (Figure 2.11) is pressed anytime during actual game play.

2.2.9.1. Selecting the “Resume Game” menu entry shall return the user to the currently running game.

2.2.9.2. Selecting the “Sound” menu entry shall display the sound menu screen.

2.2.9.3. Selecting the “Quit Game” menu entry shall exit the currently running game and display the main menu screen.

Figure 2.7 Pause Menu Screen

2.2.10. Menu Screen: Character Selection Screen (Figure 2.8) The character selection screen shall be displayed when the “Start Game” menu entry is selected from the main menu screen.

2.2.10.1. Selecting a character shall display the game play screen and begin a new game with the selected character.

2.2.10.2. Pressing the BACK button (Figure 2.11) shall display the main menu screen.

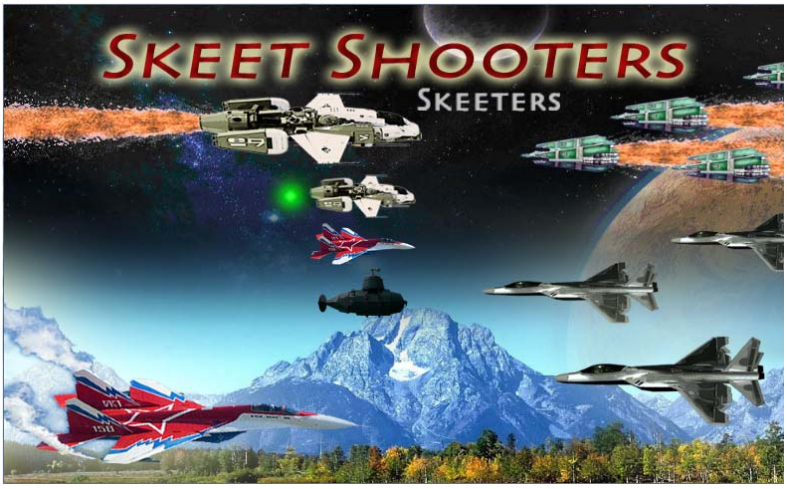


Figure 2.8 Character Selection Screen

2.2.11. Game Play Screen (Figure 2.9) The game play screen shall be displayed once a character is selected from the character selection screen.

2.2.11.1. The player's status bar (Figure 2.9 bottom of image) shall display the following:

- Player's life points.
    - One green orb = one life point
    - When no life points remain the game shall be over.
- Percentage of shield power remaining.
- Amount of bonus time remaining in the bonus gauge.
    - One red orb = one second
- Player's score.

2.2.11.2. Enemies shall be displayed flying from right to left across the game play screen.

2.2.11.3. Each time a new enemy is to be drawn on the game play screen, a random velocity and flying height shall be given to that enemy.

2.2.11.4. If an enemy runs into the player's character, the following shall happen:

- The player's life shall be decreased by one unit.
- The enemy shall disappear.
- Explosion fire and smoke shall be displayed at the point where the enemy was.

2.2.11.5. If an enemy flies off screen, the enemy shall disappear.

2.2.11.6. Enemies shall randomly shoot bullets.

2.2.11.7. When an enemy bullet is shot, the bullet shall start at the (X, Y) position of the enemy that shot it and move as follows:

- The enemy bullet shall move left, along the X axis, while keeping the same Y coordinate.
- The enemy bullet shall move at a velocity slightly faster than the enemy that shot it.

2.2.11.8. If an enemy bullet hits the player's character, the player's life shall be decreased by one unit and the following shall happen:

- The enemy bullet shall disappear
- A small explosion fire shall be displayed at the point where the enemy bullet hit the player's character.

2.2.11.9. If an enemy bullet flies off screen, the bullet shall disappear.

2.2.11.10. Moving the Left Stick (Figure 2.11) up, down, left or right shall move the player's character accordingly.

2.2.11.11. Pressing the A button (Figure 2.11) shall make the player's character shoot a bullet.

2.2.11.12. When a player bullet is shot, the bullet shall start at the X,Y position of the player's character and move as follows:

- The bullet shall move right, along the X axis, while keeping the same Y coordinate.
- The bullet shall move at a velocity slightly faster than the player's current velocity.

2.2.11.13. If an enemy is hit by one of the player's bullets, the following shall happen:

- The enemy shall disappear.
- Explosion fire and smoke shall be displayed at the point where the enemy was hit.
- The player's score shall be increased.

2.2.11.14. If a bullet flies off-screen, the bullet shall disappear.

2.2.11.15. Holding the X button (Figure 2.11) shall display a shield surrounding the player's character.

2.2.11.16. If an enemy is hit by the shield, the following shall happen:

- The enemy shall disappear.
- Explosion fire and smoke shall be displayed at the point where the enemy was hit.
- The player's score shall be increased.
- Shield power shall be decreased.

2.2.11.17. If an enemy bullet hits the shield, the following shall happen:

- Enemy bullet shall disappear.
- Shield power shall be decreased.

2.2.11.18. Enemies shall randomly drop items, called "power ups" (Figure 2.12), when killed by the player's shield or by one of the player's bullets.

2.2.11.19. The initial position of the power up shall be the same X,Y coordinate as the enemy that dropped the power up.

2.2.11.20. Once dropped, the power up shall be displayed as falling from its initial position to a position off screen.

2.2.11.21. If the power up falls off-screen it shall disappear.

2.2.11.22. When the player intersects a power up, the player shall gain the attribute of that power up.

2.2.11.23. The power up names and their corresponding attributes shall be as follows:

- Life Up (green)
  - Intersecting a Life Up shall add one life point to the player's life points.
- Shield Up (blue)
  - Intersecting a Shield Up shall replenish the player's shield back to 100%.
- Bonus Up (red)
  - Intersecting a Bonus Up shall give the player one of the following abilities:
    - Bullet Speed Up
      - The velocity at which player's bullets move shall be incremented.
    - Points Up
      - The points earned per kill shall be incremented.
    - Spray Shot Up
      - The number of bullets shot when the A button is pressed shall be incremented.

2.2.11.24. The collected Bonus Up ability shall be displayed, for a few seconds, at the position where the score is usually displayed.

2.2.11.25. When the player intersects a Bonus Up, the bonus gauge shall fill up (15 seconds total) and the following shall happen:

- The bonus gauge time shall decrease every second for 15 seconds.

- When another Bonus Up is intersected, while the bonus gauge has time remaining, the bonus gauge shall fill up and the Bonus Up ability shall be added on to any previously collected abilities.
- When the bonus gauge reaches zero, all of the collected Bonus Up abilities shall be removed from the player.

2.2.11.26. Pressing the START button (Figure 2.11) while the game play screen is displayed shall display the pause menu screen.

Figure 2.9 Game Play Screen

2.2.12. Game Tutorial Screen (Figure 2.10) The game tutorial screen shall be displayed when the "Tutorial" menu entry is selected from the options menu screen.

2.2.12.1. The player's status bar (Figure 2.10 bottom of image) shall display the following:

- Player's life points.
  - One green orb = one life point
- Percentage of shield power remaining.
- Amount of bonus time remaining in the bonus gauge.
  - One red orb = one second
- Player's score.

2.2.12.2. An introduction message (Figure 2.10) shall be displayed introducing the user to the game tutorial and prompting the user to press the B button, on the Xbox 360 controller (Figure 2.11), to continue.

2.2.12.3. Once the B button is pressed, a message shall be displayed explaining to the user that moving the Left Stick, on the Xbox 360 controller (Figure 2.11), moves the player accordingly.

- The message shall continue being displayed until the Left stick is moved.

2.2.12.4. Once the Left stick is moved, the player shall be moved in the corresponding direction until the Left stick is released.

- Moving the Left stick UP, DOWN, LEFT or RIGHT, shall move the player's character accordingly.

2.2.12.5. The tutorial shall wait a few second, after the initial movement of the Left stick, before displaying the next message.

2.2.12.6. Once a few seconds have passed, a message shall be displayed prompting the user to press the A button, on the Xbox 360 controller (Figure 2.11), to shoot a bullet

2.2.12.7. Pressing the A button shall make the player's character shoot a bullet.

2.2.12.8. When a bullet is shot, the bullet shall start at the X,Y position of the player's character and move as follows:

- The bullet shall move right, along the X axis, while keeping the same Y coordinate.
- The bullet shall move at a velocity slightly faster than the player's current velocity.
- If a bullet flies off-screen, the bullet shall disappear.

2.2.12.9. Once the bullet has flown off-screen, a message shall be displayed prompting the user to shoot an enemy with a bullet.

2.2.12.10. A maximum of two enemies shall be displayed on the screen at any given time during the rest of the tutorial.

2.2.12.11. Enemies shall be displayed flying from right to left across the game play screen.

2.2.12.12. Each time a new enemy is to be drawn, a random velocity and flying height shall be given to that enemy.

2.2.12.13. If an enemy runs into the player's character the following shall happen:

- The player's life shall be decreased by one unit.
    - If the player's life points reach zero, then a message shall be displayed explaining to the user that they have run out of life points and prompting the user to press the B button to continue.
        - Pressing the B button shall do the following:
            - Refill the player's life points.
            - Continue with the tutorial.
- The enemy shall disappear.
- Explosion fire and smoke shall be displayed at the point where the enemy was.

2.2.12.14. If an enemy flies off screen, the enemy shall disappear and a new enemy shall be drawn.

2.2.12.15. Once an enemy is hit by one of the player's bullets, the following shall happen:

- The enemy shall disappear.
- Explosion fire and smoke shall be displayed at the point where the enemy was hit.
- The player's score shall be increased.
- A message shall be displayed explaining where the player's score is displayed and how to increase the player's score.
    - The message shall prompt the user to press the B button to continue.

2.2.12.16. Once the B button has been pressed, a message shall be displayed explaining where the player's life points are displayed.

- The message shall also prompt the user to press the B button to continue.

2.2.12.17. Once the B button has been pressed a message shall be displayed explaining that when the player gets hit by an enemy or enemy bullet, the players life points shall decrease by one unit.

- The message shall also prompt the user to press the B button to continue.

2.2.12.18. Once the B button has been pressed, a message shall be displayed prompting the user to hold the X button, on the Xbox 360 controller (Figure 2.11), to activate their player's shield.

- Holding the X button shall display a shield surrounding the player's character.

2.2.12.19. Once the B button is pressed, a message shall be displayed explaining that the shield can be used to prevent enemies and their bullets from damaging the player.

- The message shall also prompt the user to press the B button to continue.

2.2.12.20. Once the B button is pressed, a message shall be displayed explaining where the player's shield power is displayed and how the shield power decreases each time the player's shield is hit by an enemy or enemy bullet.

- The message shall also prompt the user to press the B Button to continue.

2.2.12.21. Once the B button is pressed, a message shall be displayed prompting the user to hit an enemy with the shield activated.

2.2.12.22. Once an enemy is hit by the shield, the following shall happen:

- The enemy shall disappear.
- Explosion fire and smoke shall be displayed at the point where the enemy was hit.
- The player's score shall be increased.
- Shield power shall be decreased.
- A message shall be displayed explaining that the player's score increases when an enemy is hit by the player's shield.

- The message shall also prompt the user to press the B button to continue.

2.2.12.23. Once the B button is pressed, a message shall be displayed prompting the user to hit another enemy with a bullet or the shield.

2.2.12.24. When an enemy is hit in this way, the following shall happen:

- A power up shall be displayed at the point where the enemy was killed.
- A message shall be displayed explaining that the dropped item is referred to as a "power up".
    - The message shall also prompt the user to press the B button to continue.

2.2.12.25. Once the B button is pressed, a message shall be displayed explaining that power ups are randomly dropped by enemies when killed.

- The message shall also prompt the user to press the B button to continue.

2.2.12.26. Once the B button is pressed the following shall happen:

- A life up (green orb) shall be displayed.
- A message shall be displayed explaining that this power up is called a life up.
    - The message shall also explain that this power up adds one life point to the player's current life points
    - The message shall also prompt the user to press the B button to continue.

2.2.12.27. Once the B buton is pressed the following shall happen:

- A shield up (blue orb) shall be displayed.
- A message shall be displayed explaining that this power up is called a shield up.
    - The message shall also explain that this power up replenishes the player's shield power back to 100%.
    - The message shall also prompt the user to press the B button to continue.

2.2.12.28. Once the B button is pressed the following shall happen:

- A bonus up (red orb) shall be displayed.
- A message shall be displayed explaining that this power up is called a bonus up.
- The message shall also explain that this power up will give the player one of three abilities:
    - Bullet Speed Up - The velocity at which player's bullets move shall be incremented.
    - Points Up - The points earned per kill shall be incremented.
    - Spray Shot Up - The number of bullets shot when the A button is pressed shall be incremented.
- The message shall also prompt the user to press the B button to continue.

2.2.12.29. Once the B button is pressed, a message shall be displayed explaining where the bonus gauge is displayed.

- The message shall also prompt the user to press the B button to continue.

2.2.12.30. Once the B button is pressed, a message shall be displayed explaining that whenever the player intersects a bonus up, and the correct answer is selected, the player's bonus gauge will fill.

- The message shall also prompt the user to press the B button to continue.

2.2.12.31. Once the B button is pressed, a message shall be displayed explaining where the collected bonus up effect is displayed.

- The message shall also prompt the user to press the B button to continue.

2.2.12.32. Once the B button is pressed, a message shall be displayed explaining that the player's bonus gauge decreases over time.

- The message shall also prompt the user to press the B button to continue.

2.2.12.33. Once the B button is pressed, a message shall be displayed explaining that when the bonus gauge runs out, the player will lose all bonus effects that have been collected so far.

- The message shall also prompt the user to press the B button to continue.

2.2.12.34. Once the B button is pressed, a message shall be displayed explaining that it is possible to collect multiple bonus effects by keeping the bonus gauge active.

- The message shall also prompt the user to press the B button to continue.

2.2.12.35. Once the B button is pressed, a message shall be displayed explaining that multiple bonus effects can be collected by collecting another bonus up before the bonus gauge becomes empty.

- The message shall also prompt the user to press the B button to continue.

2.2.12.36. Once the B button is pressed, a message shall be displayed ending the tutorial.

- The message shall also prompt the user to press the B button to continue.

Figure 2.10 Game Tutorial Screen

Figure 2.11 XBox 360 Controller Layout

Figure 2.12 Power Up Images

3. Design Solution

The XBox 360 along with the testing computer running Microsoft Visual Studio 2008 and Game Studio 3.1 is used to complete this project. The XBox 360 must be connected to a video display device for testing purposes and must be on the same network as the testing computer. A Madcatz wireless adapter is used to connect the XBox 360 to the same wireless network in which the testing computer is connected. The game program is written using C# and XNA framework with the above mentioned editing software. All major classes are described below.

3.1. Miscellaneous Classes (Figure 3.1)

3.1.1. SreenManager.cs

A screen manager class was added to manage the transitioning between screens. The screen manager is a component which manages one or more GameScreen.cs instances. It maintains a stack of screens, calls their Update and Draw methods at the appropriate times, and automatically routes input to the topmost active screen.

3.1.2. AudioManager.cs

An audio manager class was added to manage the playback of the various game sounds and music.

3.1.3. InputState.cs

An input state class was added to act as a helper for reading input from the XBox 360 controller. This class tracks both the current and previous state of both input devices, and implements query methods for high level input actions such as "move up through the menu" or "pause the game".

3.1.4. PlayerIndexEventArgs.cs

A player index event class was added to customize event argument to include the index of the player who triggered the event.

3.1.5. Program.cs

The program class is the main entry point of the application.

3.1.6. SkeetShooters.cs

The skeet shooters class initializes an instance of ScreenManager.cs and AudioManager.cs.

3.2. Particle System Classes (Figure 3.2)

3.2.1. ParticleSystem.cs

ParticleSystem.cs is an abstract class that provides the basic functionality to create a particle effect. Different subclasses have different effects, such as fire, explosions, and plumes of smoke.

3.2.2. Particle.cs

Particles.cs represents a single particle used to create an effect. Each effect is comprised of many of these particles. Particles have basic physical properties, such as position, velocity, acceleration, and rotation.

3.2.3. ExplosionParticleSystem.cs

ExplosionParticleSystem.cs is a specialization of ParticleSystem.cs that creates a fiery explosion.

3.2.4. ExplosionSmokeParticleSystem.cs

ExplosionSmokeParticleSystem.cs is a specialization of ParticleSystem.cs that creates a circular pattern of smoke.

3.3. Menu Screen Classes (Figure 3.4)

3.3.1. MenuScreen.cs

The menu screen class was added as a base class for all screen classes that contain a menu of options.

3.3.2. MenuEntry.cs

The menu entry class represents a single entry in a menu screen.

3.3.3. MainMenuScreen.cs

The main menu screen class inherits MenuScreen.cs and is displayed initially at the beginning of the game.

3.3.4. OptionsMenuScreen.cs

The option menu screen class inherits MenuScreen.cs and is displayed when the “Options” menu entry is selected from the main menu screen.

3.3.5. PauseMenuScreen.cs

The pause menu screen class inherits MenuScreen.cs and is displayed whenever the game is paused.

3.3.6. SkeetScreen.cs

The skeeter selection screen class inherits MenuScreen.cs and is displayed when the “Start Game” menu entry is selected from the main menu screen.

3.3.7. SoundMenuScreen.cs

The sound menu screen class inherits MenuScreen.cs and is displayed when the “Sound” menu entry is selected from the options menu screen or the pause menu screen.

3.3.8. MusicVolumeScreen.cs

The music volume menu screen class inherits MenuScreen.cs and is displayed when the “Music Volume” menu entry is selected from the sound menu screen.

3.3.9. SfxVolumeScreen.cs

The SFX volume menu screen class inherits MenuScreen.cs and is displayed when the “SFX Volume” menu entry is selected from the sound menu screen.

3.4. Game Screen Classes (Figure 3.5 and 3.6)

3.4.1. GameScreen.cs

The game screen class was added as a base class for every screen to inherit.
An instance of the GameScreen.cs is a single layer that has update and draw logic, and which can be combined with other layers to build up a complex menu system.

3.4.2. BackgroundScreen.cs

The background screen sits behind all of the other menu screens. It draws a background image that remains fixed in place regardless of whatever transitions the screens on top of it may be doing.

3.4.3. LoadingScreen.cs

The loading screen coordinates transitions between the menu system and the game itself.

3.4.4. MessageBoxScreen.cs

A popup message box screen, used to display "are you sure?" confirmation messages.

3.4.5. GameTutorial.cs

This screen is displayed when the “Tutorial” menu entry is selected from the options menu screen. This class contains all of the logic for running the interactive game tutorial.

3.4.6. GameplayScreen.cs

This screen is displayed when a Skeeter is selected from the skeet selection screen. This class contains all of the logic for running the actual game play.

3.4.7. GameObject.cs

An instance of this class represents a single game object in GameTutorial.cs and GameplayScreen.cs.

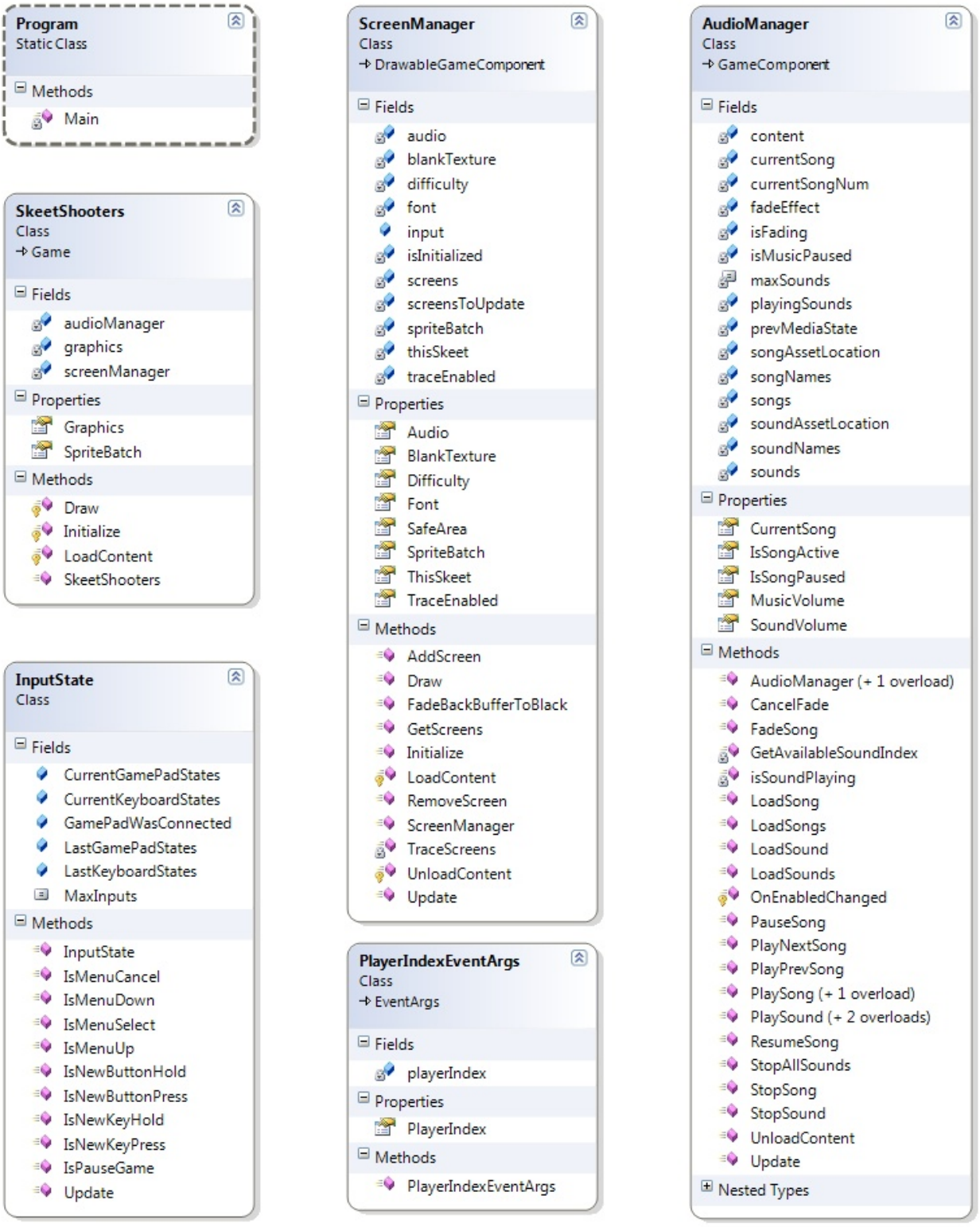


Figure 3.1 Miscellaneous Classes

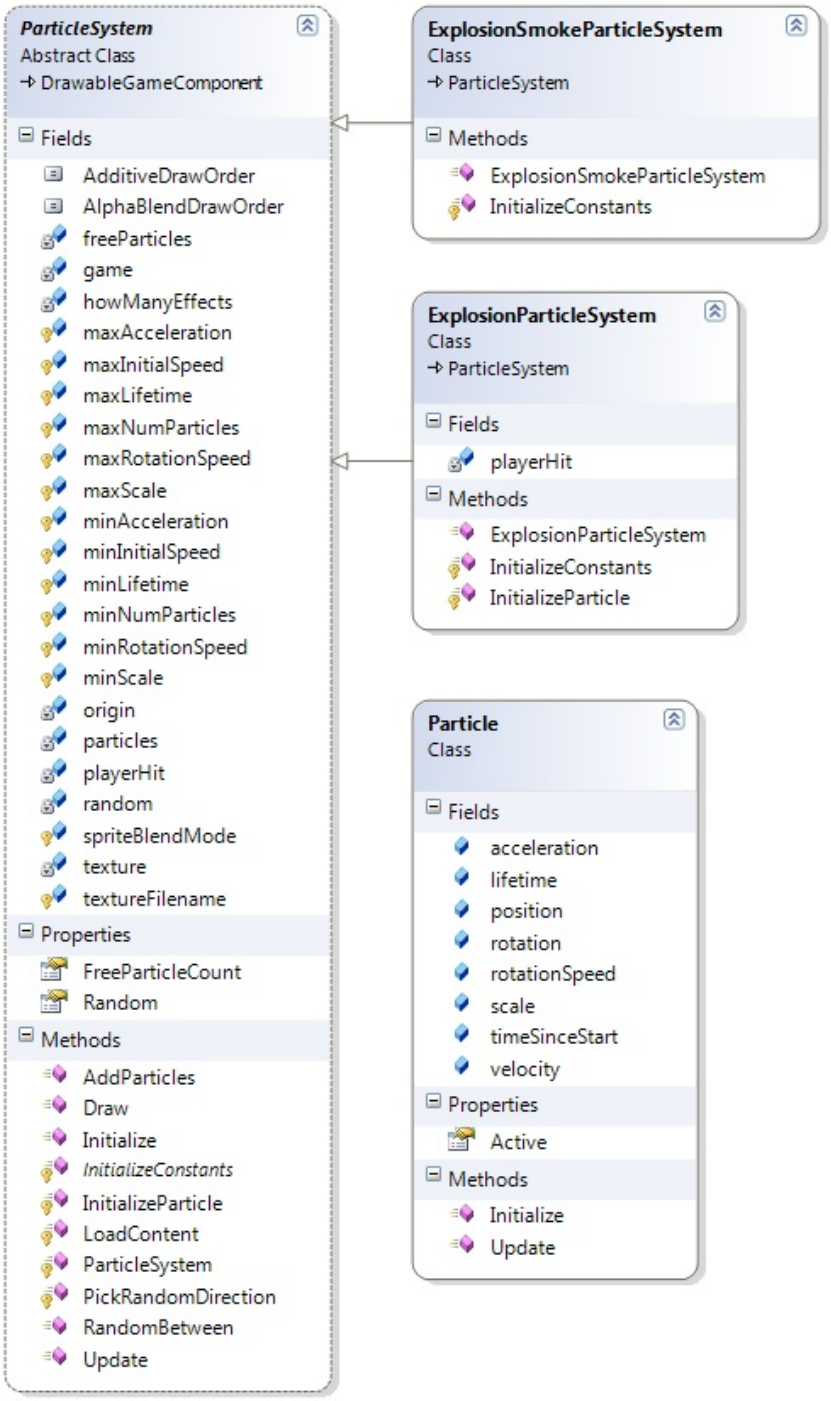


Figure 3.2 Particle System Classes

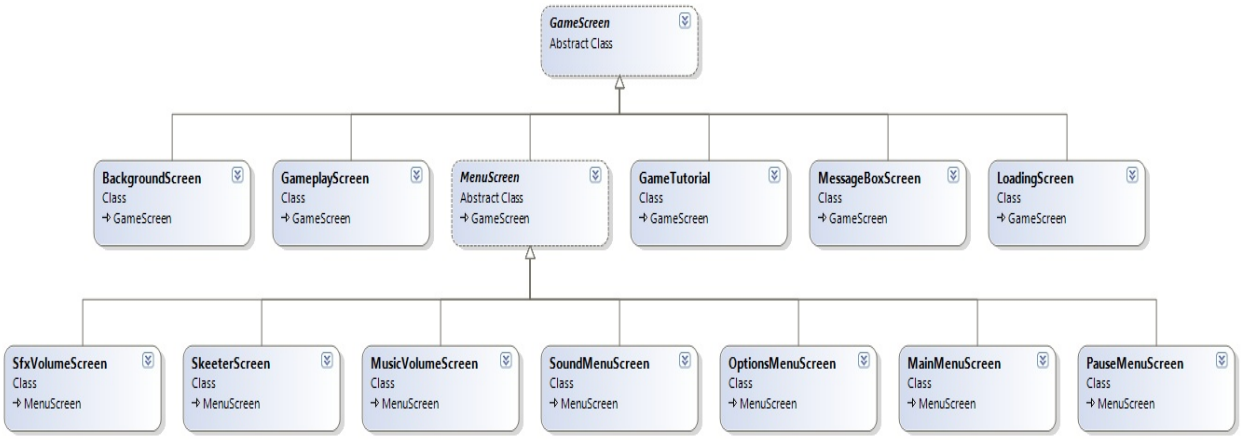


Figure 3.3 Game and Menu Screen Classes (Overview)

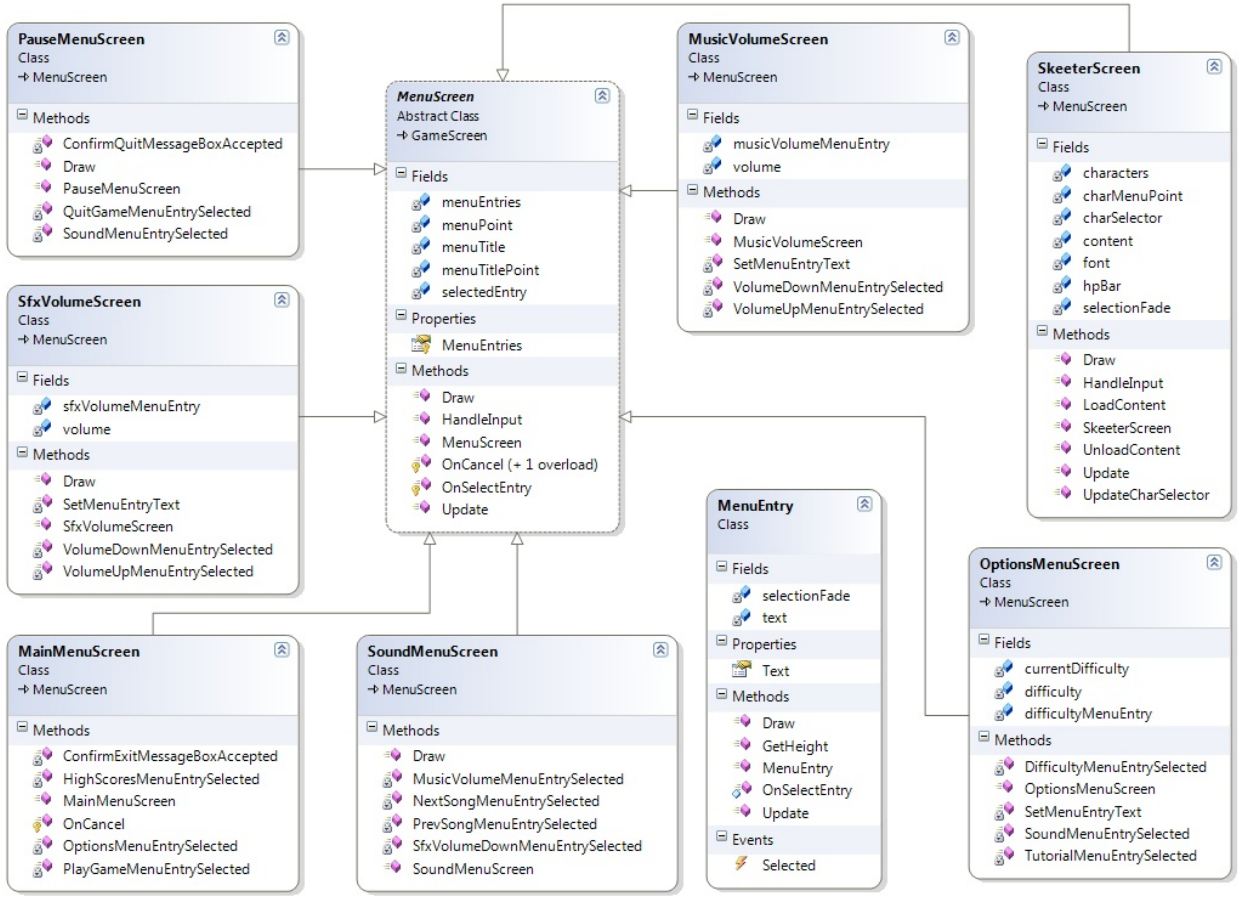


Figure 3.4 Menu Screen Classes

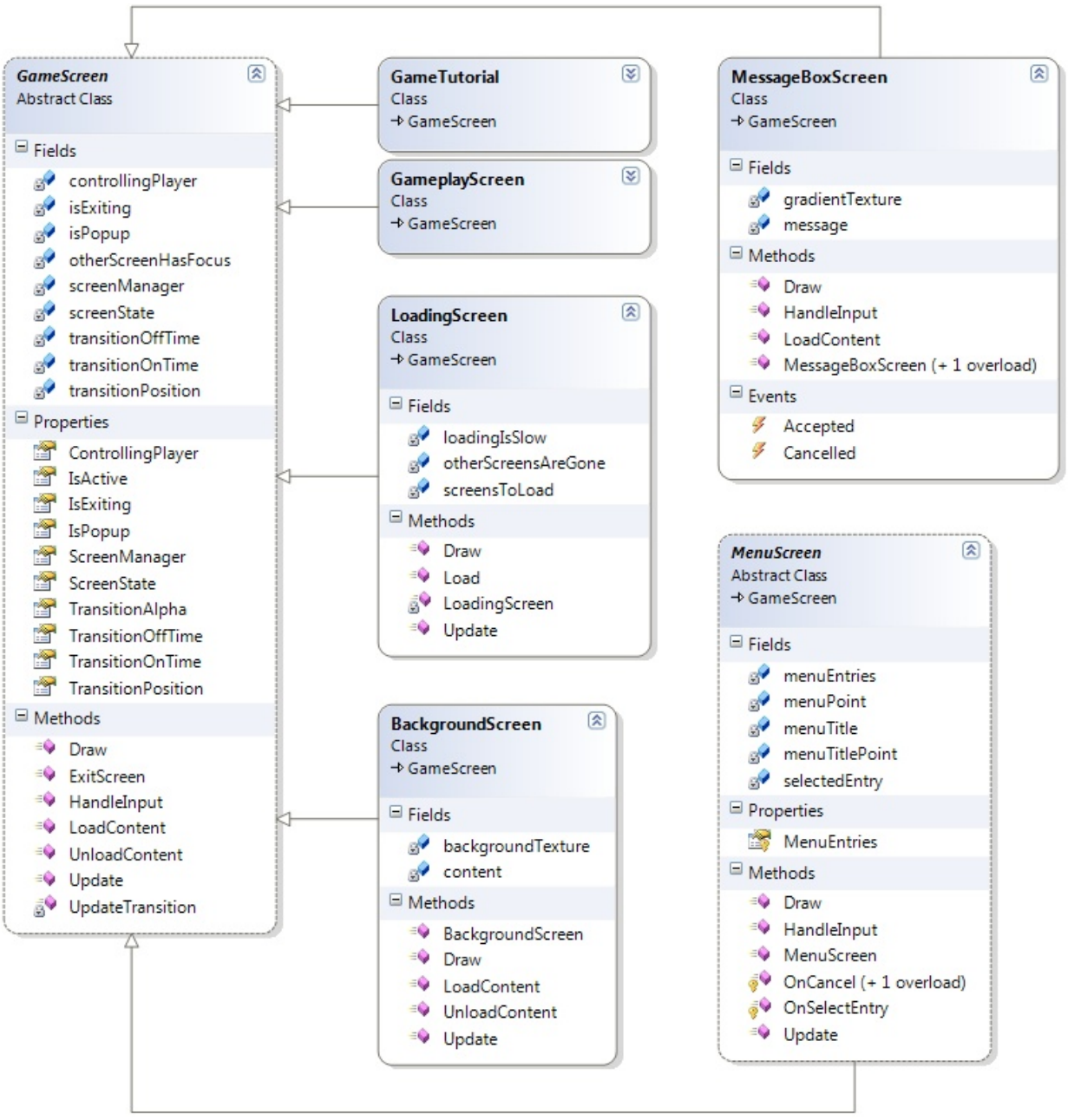


Figure 3.5 Game Screen Classes

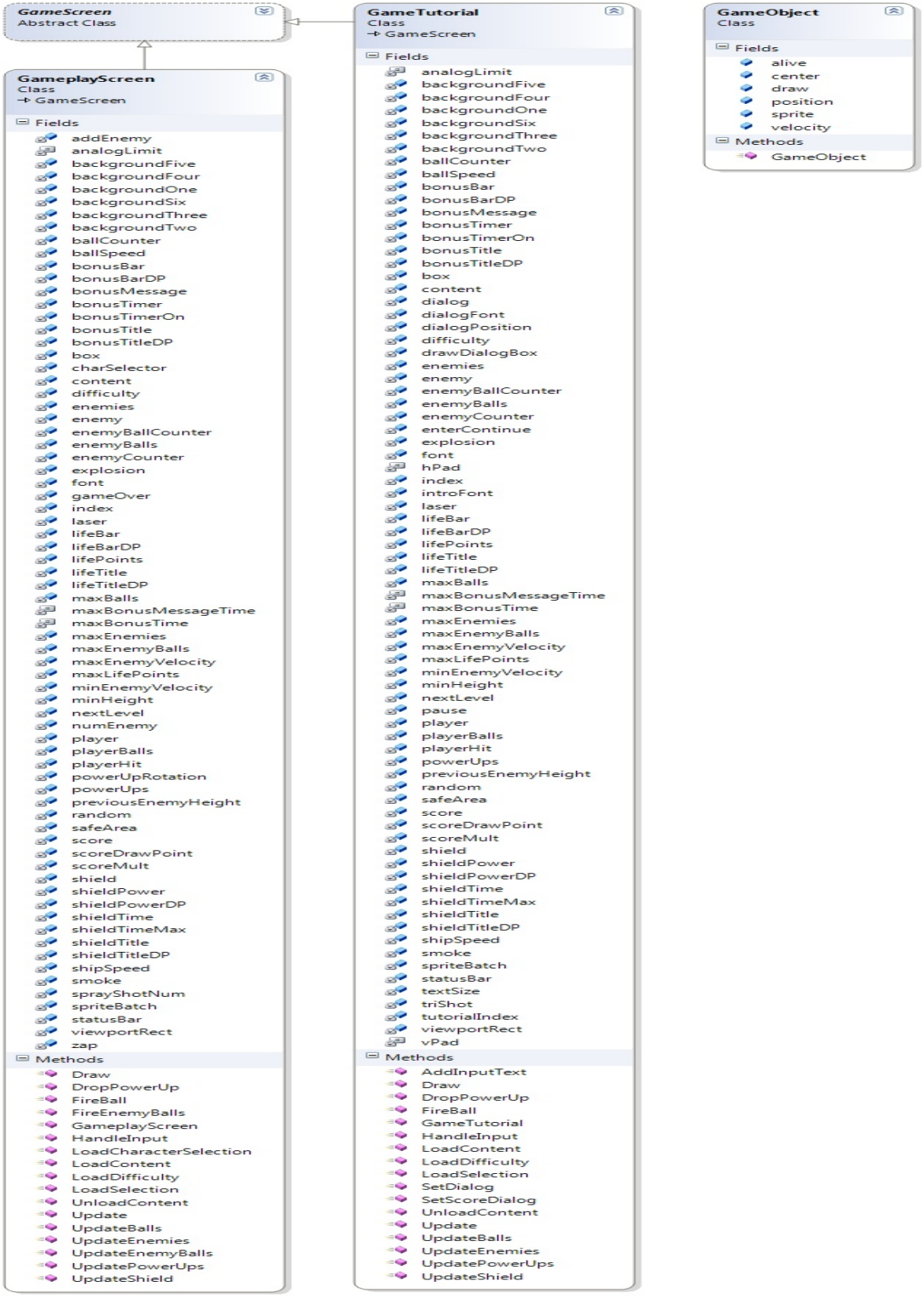


Figure 3.6 Game Play and Game Tutorial Classes

4. Implementation

The main entry point for the game application is in Program.cs. This class creates an instance of the SkeetShooters.cs class and then calls the Run() method on the created instance, as shown below.

```
static void Main(string[] args)
{
   using (SkeetShooters game = new SkeetShooters())
   {
         game.Run();
   }
}
```

The main class for the game application is SkeetShooters.cs. This class creates an instance of AudioManager.cs and ScreenManager.cs and then adds a new instance of the MainMenuScreen.cs to the ScreenManager.cs instance, as shown below.

```
public SkeetShooters()
{
   // initialize graphics device
   graphics = new GraphicsDeviceManager(this);

   Content.RootDirectory = "Content";

   // initialize audio manager component
   audioManager = new AudioManager(this);
   Components.Add(audioManager); // add component

   // initialize screen manager component.
   screenManager = new ScreenManager(this, audioManager);

   // activate the first screens.
   screenManager.AddScreen(new BackgroundScreen(), null);
   screenManager.AddScreen(new MainMenuScreen(), null);

   Components.Add(screenManager); // add component
}
```

The class that manages one or more GameScreen.cs instances is ScreenManager.cs. It maintains a stack of screens, calls their Update and Draw methods at the appropriate time and automatically routes input to the topmost active screen.

AddScreen method is used to add a new screen to the screen manager.

```
public void AddScreen(GameScreen screen, PlayerIndex? controllingPlayer)
{
   screen.ControllingPlayer = controllingPlayer;
   screen.ScreenManager = this;
   screen.IsExiting = false;
```

```
    // If we have a graphics device, tell the screen to load content.
    if (isInitialized)
    {
          screen.LoadContent();
    }
    screens.Add(screen);

}
```

RemoveScreen method is used to remove a screen from the screen manager.

```
public void RemoveScreen(GameScreen screen)
{
    // If we have a graphics device, tell the screen to unload content.
    if (isInitialized)
    {
          screen.UnloadContent();
    }

    screens.Remove(screen);
    screensToUpdate.Remove(screen);
}
```

The class that is the base class for all of the game screens is GameScreen.cs. A screen is a single layer that has update and draw logic, and which can be combined with other layers to build a complex menu system. Each of the game's screens inherits GameScreen.cs.

UpdateTransition method is used as a helper for updating the screen transition position.

```
bool UpdateTransition(GameTime gameTime, TimeSpan time, int direction)
{
    // How much should we move by?
    float transitionDelta;

    if (time == TimeSpan.Zero)
          transitionDelta = 1;
    else
          transitionDelta =
                (float)(gameTime.ElapsedGameTime.TotalMilliseconds /
                time.TotalMilliseconds);

    // Update the transition position.
    transitionPosition += transitionDelta * direction;

    // Did we reach the end of the transition?
    if (((direction < 0) && (transitionPosition <= 0)) ||
         ((direction > 0) && (transitionPosition >= 1)))
    {
          transitionPosition =
          MathHelper.Clamp(transitionPosition, 0, 1);
          return false;
    }

    // Otherwise we are still busy transitioning.
```

```
      return true;
}
```

The class that is the base class for all game screens containing a menu of options is MenuScreen.cs. Each of the game's menu screens inherits MenuScreen.cs.

The Update method updates the menu screen and its menu entries.

```
public override void Update(GameTime gameTime, bool otherScreenHasFocus,
   bool coveredByOtherScreen)
{
   base.Update(gameTime, otherScreenHasFocus, coveredByOtherScreen);

   // Update each nested MenuEntry object.
   for (int i = 0; i < menuEntries.Count; i++)
   {
         bool isSelected = IsActive && (i == selectedEntry);
         menuEntries[i].Update(this, isSelected, gameTime);
   }

}
```

MainMenuScreen.cs inherits MenuScreen.cs and is where the main menu screen logic is implemented.

The class constructor initializes the menu entries and hooks up an event handler to each menu entry.

```
public MainMenuScreen()
   : base("Main Menu")
{
   // Create our menu entries.
   MenuEntry playGameMenuEntry = new MenuEntry("Start Game");
   MenuEntry optionsMenuEntry = new MenuEntry("Options");
   MenuEntry exitMenuEntry = new MenuEntry("Exit");

   // Hook up menu event handlers.
   playGameMenuEntry.Selected += PlayGameMenuEntrySelected;
   optionsMenuEntry.Selected += OptionsMenuEntrySelected;
   exitMenuEntry.Selected += OnCancel;

   // Add entries to the menu.
   MenuEntries.Add(playGameMenuEntry);
   MenuEntries.Add(optionsMenuEntry);
   MenuEntries.Add(exitMenuEntry);
}
```

OptionsMenuScreen.cs inherits MenuScreen.cs and is where the options menu screen logic is implemented.

The class constructor initializes the menu entries and hooks up an event handler to each menu entry.

```
public OptionsMenuScreen()
    : base("Options")
{
    // Create our menu entries.
    difficultyMenuEntry = new MenuEntry(string.Empty);

    SetMenuEntryText();

    MenuEntry tutorialMenuEntry = new MenuEntry("Tutorial");
    MenuEntry soundMenuEntry = new MenuEntry("Sound");
    MenuEntry backMenuEntry = new MenuEntry("Back");

    // Hook up menu event handlers.
    difficultyMenuEntry.Selected += DifficultyMenuEntrySelected;
    tutorialMenuEntry.Selected += TutorialMenuEntrySelected;
    soundMenuEntry.Selected += SoundMenuEntrySelected;
    backMenuEntry.Selected += OnCancel;

    // Add entries to the menu.
    MenuEntries.Add(difficultyMenuEntry);
    MenuEntries.Add(tutorialMenuEntry);
    MenuEntries.Add(soundMenuEntry);
    MenuEntries.Add(backMenuEntry);
}
```

SoundMenuScreen.cs inherits MenuScreen.cs and is where the sound menu screen logic is implemented.

The class constructor initializes the menu entries and hooks up an event handler to each menu entry.

```
public SoundMenuScreen(bool isPopup)
    : base("Sound")
{
    IsPopup = isPopup;

    MenuEntry musicVolumeMenuEntry = new MenuEntry("Music Volume");
    MenuEntry sfxVolumeDownMenuEntry = new MenuEntry("SFX Volume");
    MenuEntry nextSongMenuEntry = new MenuEntry("Next Song");
    MenuEntry prevSongMenuEntry = new MenuEntry("Prev Song");
    MenuEntry backMenuEntry = new MenuEntry("Back");

    // Hook up menu event handlers.
    musicVolumeMenuEntry.Selected += MusicVolumeMenuEntrySelected;
    sfxVolumeDownMenuEntry.Selected += SfxVolumeDownMenuEntrySelected;
    nextSongMenuEntry.Selected += NextSongMenuEntrySelected;
    prevSongMenuEntry.Selected += PrevSongMenuEntrySelected;
    backMenuEntry.Selected += OnCancel;
```

```
    // Add entries to the menu.
    MenuEntries.Add(musicVolumeMenuEntry);
    MenuEntries.Add(sfxVolumeDownMenuEntry);
    MenuEntries.Add(nextSongMenuEntry);
    MenuEntries.Add(prevSongMenuEntry);
    MenuEntries.Add(backMenuEntry);
}
```

MusicVolumeScreen.cs inherits MenuScreen.cs and is where the music volume menu screen logic is implemented.

The class constructor initializes the menu entries and hooks up an event handler to each menu entry.

```
public MusicVolumeScreen(float musicVolume, bool isPopup)
    : base("Music")
{
    IsPopup = isPopup;

    // Create our menu entries.
    musicVolumeMenuEntry = new MenuEntry(string.Empty);
    // convert volume to percent
    this.volume = (int)(musicVolume * 100);
    if (volume % 10 != 0)
          volume = (10 - (volume % 10)) + volume;

    SetMenuEntryText();

    MenuEntry volumeUpMenuEntry = new MenuEntry("Volume Up");
    MenuEntry volumeDownMenuEntry = new MenuEntry("Volume Down");
    MenuEntry backMenuEntry = new MenuEntry("Back");

    // Hook up menu event handlers.
    volumeUpMenuEntry.Selected += VolumeUpMenuEntrySelected;
    volumeDownMenuEntry.Selected += VolumeDownMenuEntrySelected;
    backMenuEntry.Selected += OnCancel;

    // Add entries to the menu.
    MenuEntries.Add(musicVolumeMenuEntry);
    MenuEntries.Add(volumeUpMenuEntry);
    MenuEntries.Add(volumeDownMenuEntry);
    MenuEntries.Add(backMenuEntry);
}
```

SfxVolumeScreen.cs inherits MenuScreen.cs and is where the sound effects volume menu screen logic is implemented.

The class constructor initializes the menu entries and hooks up an event handler to each menu entry.

```
public SfxVolumeScreen(float sfxVolume, bool isPopup)
```

```
    : base("Sound FX")
{
    IsPopup = isPopup;

    // Create our menu entries.
    sfxVolumeMenuEntry = new MenuEntry(string.Empty);

    // convert volume to percent
    volume = (int)(sfxVolume * 100);
    if(volume % 10 != 0)
          volume = (10 - (volume % 10)) + volume;

    SetMenuEntryText();

    MenuEntry volumeUpMenuEntry = new MenuEntry("Volume Up");
    MenuEntry volumeDownMenuEntry = new MenuEntry("Volume Down");
    MenuEntry backMenuEntry = new MenuEntry("Back");

    // Hook up menu event handlers.
    volumeUpMenuEntry.Selected += VolumeUpMenuEntrySelected;
    volumeDownMenuEntry.Selected += VolumeDownMenuEntrySelected;
    backMenuEntry.Selected += OnCancel;

    // Add entries to the menu.
    MenuEntries.Add(sfxVolumeMenuEntry);
    MenuEntries.Add(volumeUpMenuEntry);
    MenuEntries.Add(volumeDownMenuEntry);
    MenuEntries.Add(backMenuEntry);
}
```

PauseMenuScreen.cs inherits MenuScreen.cs and is where the pause menu screen logic is implemented.

The class constructor initializes the menu entries and hooks up an event handler to each menu entry.

```
public PauseMenuScreen()
    : base("Paused")
{
    // Flag that there is no need for the game to transition
    // off when the pause menu is on top of it.
    IsPopup = true;

    // Create our menu entries.
    MenuEntry resumeGameMenuEntry = new MenuEntry("Resume Game");
    MenuEntry soundMenuEntry = new MenuEntry("Sound");
    MenuEntry quitGameMenuEntry = new MenuEntry("Quit Game");

    // Hook up menu event handlers.
    resumeGameMenuEntry.Selected += OnCancel;
    soundMenuEntry.Selected += SoundMenuEntrySelected;
    quitGameMenuEntry.Selected += QuitGameMenuEntrySelected;
```

```
        // Add entries to the menu.
        MenuEntries.Add(resumeGameMenuEntry);
        MenuEntries.Add(soundMenuEntry);
        MenuEntries.Add(quitGameMenuEntry);
    }
```

5. Conclusion

The Skeet Shooters Windows game was successfully ported to run on the XBox 360 video game console.

Playing the game on the XBox 360 requires that it and the Windows PC be connected to the same network with access to the Internet. The XBox 360 requires a number of network ports to be “open” in order to access the Internet (see user manual). Testing the game on the XBox 360 in the computer science lab proved to be quite difficult because many of the needed ports have been “closed” throughout the entire campus.

Further extensions to the game may include:

- High scores screen to display the top scores.
- More selectable characters and backgrounds.

6. References

[1] App Hub, Microsoft Corporation, 1-21-2011, <http://create.msdn.com/en-US/>.

[2] "Microsoft Visual Studio." Wikipedia, 3-30-2011, <http://en.wikipedia.org/wiki/Microsoft_Visual_Studio_2008#Visual_Studio_2008>.

[3] "Microsoft XNA." Wikipedia, 3-30-2011, <http://en.wikipedia.org/wiki/XNA_Framework>.

[4] "Xbox360." Wikipedia, 3-30-2011, <http://en.wikipedia.org/wiki/Xbox_360>.

[5] "Xbox 360 Media Server Setup." Digital Trends, 3-30-2011, <http://www.digitaltrends.com/gaming/xbox-360-media-server-setup/>.

[6] "Xbox 360 Wireless Gaming Adapter." Amazon.com: Xbox 360 Wireless Gaming Adapter: Electronics, Amazon.com, 3-30-2011, <http://www.amazon.com/Xbox-Wireless-Gaming-Adapter-Playstation-3/dp/B0032J4TF8/ref=sr_1_1?ie=UTF8&s=videogames&qid=1301497590&sr=8-1>.

7. Appendix

a. User Manual

1. Downloading and Installing Software onto Windows Based PC

1.a.1.1. Download and install Microsoft Visual Studio 2008 (or Visual C# 2008 Express Edition) from www.microsoft.com.

1.a.1.2. Download and install Microsoft XNA Game Studio 3.1 from www.apphub.com.

2. Opening ports for the XBox 360.

1.a.2.1. Xbox LIVE requires the following ports to be open:

- Port 88 (UDP)
- Port 3074 (UDP and TCP)
- Port 53 (UDP and TCP)
- Port 80 (TCP)

1.a.2.2. If you're connected to a network through your workplace or school, ask the network administrator to open the above ports used by Xbox LIVE.

3. Connecting to Xbox 360 console using XNA Game Studio 3.1

This section was taken from an MSDN tutorial found here: <http://msdn.microsoft.com/en-us/library/bb975643(v=XNAGameStudio.31).aspx>.

1.a.3.1. Step 1: Sign In to Xbox LIVE

1.a.3.1.1. Turn on your Xbox 360 console, and sign in to Xbox LIVE. At the very least, you will need a Silver Xbox LIVE membership, an XNA Creators Club premium membership, and a hard drive for your Xbox 360 console to be able to develop games for Xbox 360 using XNA Game Studio. While you are in XNA Game Studio Connect or playing an XNA Game Studio game, you need to be connected to Xbox LIVE.

1.a.3.1.2. Connections between your Xbox 360 console and XNA Game Studio require a premium membership in XNA Creators Club. To sign up, visit the XNA Creators Club Web site.

1.a.3.2. Step 2: Download XNA Game Studio Connect

1.a.3.2.1. You must download XNA Game Studio Connect from Xbox LIVE Marketplace, and install it on the Xbox 360 console. Go to the Xbox LIVE Marketplace to find XNA Game Studio Connect. To download XNA Game Studio Connect, choose one of the following options:

1.a.3.2.2. Browsing to All Games using the Guide

1.a.3.2.2.1. From the Guide, navigate to the Marketplace blade.

1.a.3.2.2.2. Select Game Marketplace, then All Games. This brings you to the All Games screen.

1.a.3.2.2.3. Once you've arrived at the All Games screen from either the Guide or NXE, use the following procedure to download XNA Game Studio Connect.

1.a.3.2.3. Downloading XNA Game Studio Connect from the All Games screen

1.a.3.2.3.1. From All Games, browse to the Genre screen, and select Other.

1.a.3.2.3.2. Scroll to XNA Creators Club, and press A.

1.a.3.2.3.3. From the XNA Creators Club pane, select All Downloads, then XNA Game Studio Connect.

1.a.3.2.3.4. Press the A controller button, and select Confirm Download to begin downloading.

1.a.3.3. Step 3: Connect Your Xbox 360 Console and Windows-Based Computer

1.a.3.3.1. When you develop games for Xbox 360, you develop them on your Windows-based computer, then transfer them over your local network to your Xbox 360 console. This requires that your Windows-based computer and Xbox 360 console share the same subnet. Most home networking layouts support this configuration. If your console and computer share a router or hub, it is likely that they share the same subnet.

1.a.3.3.2. With your computer and console on the same subnet, follow these steps to set up a connection between your computer and your console.

1.a.3.3.3. Step 3A. On Your Xbox 360 Console, Generate a Connection Key:

1.a.3.3.3.1. From the Xbox Dashboard, go to My Xbox, select Game Library, and press the A controller button.

1.a.3.3.3.2. Also, you have the option of using the Xbox Guide: select the Games tab, then Game Library, and press the A controller button.

1.a.3.3.3.3. From the Game Library, go to the Collections tab, select Community Games, and press A.

1.a.3.3.3.4. Select XNA Game Studio Connect, and press A.

1.a.3.3.3.5. Select Launch, and press A.

1.a.3.3.3.6. The XNA Game Studio Connect screen appears.

1.a.3.3.3.7. If the XNA Game Studio Connect screen displays a connection key, continue to step 3B.

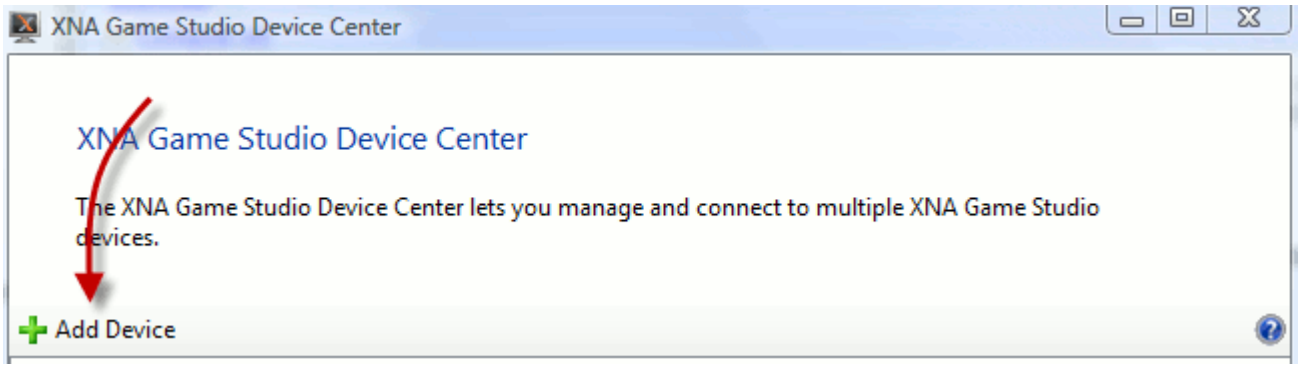
1.a.3.3.3.8. If the connection key does not appear, you can generate a new key by pressing the X controller button.

1.a.3.3.3.8.1. If the connection key does not appear, the Xbox console could already be connected to this Windows-based computer. XNA Game Studio allows multiple connection keys for multiple users on multiple computers. For more information, see Using XNA Game Studio Device Center. To add a new connection key to the list of connection keys recognized by this Xbox 360 console, press X. To reset all connection keys and generate a new connection key to connect to this Xbox 360 console, press Y.

1.a.3.3.4. Step 3B. On Your Windows-Based Computer, Enter the Connection Key and Initiate the Connection

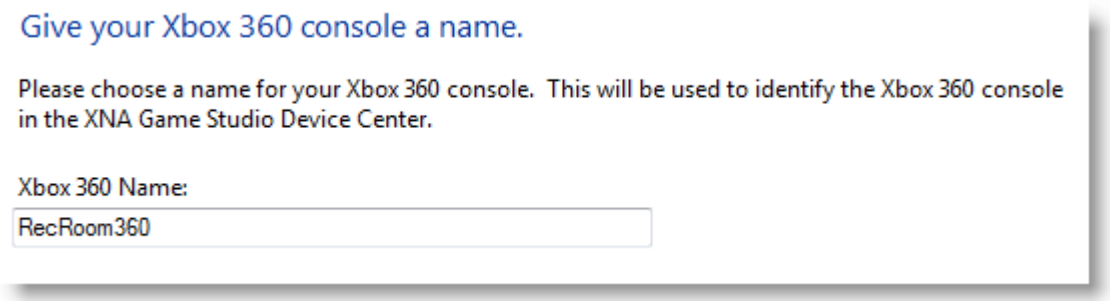
1.a.3.3.4.1. From the Start menu, select Programs, select XNA Game Studio 3.1, and launch the XNA Game Studio Device Center.

1.a.3.3.4.2. Click Add Device.



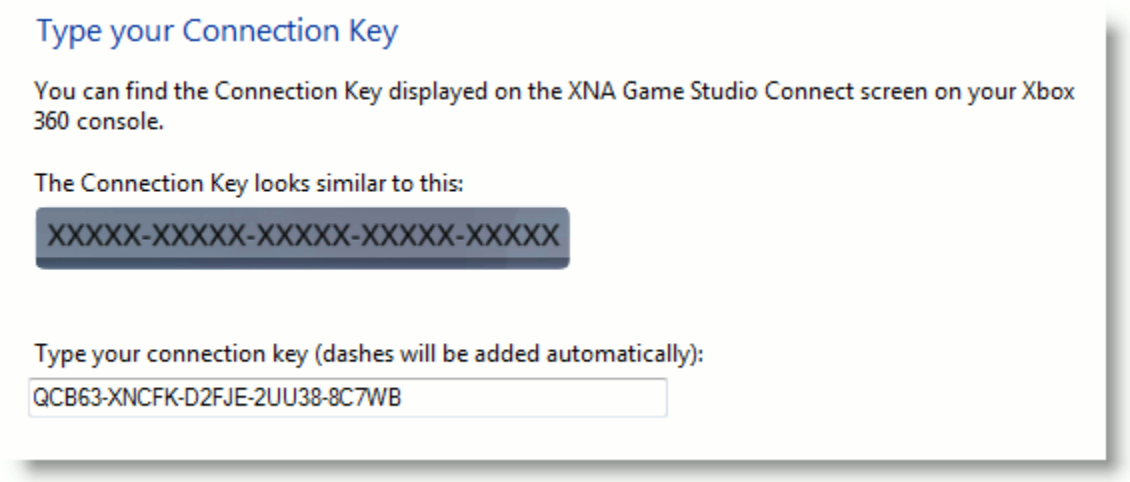
1.a.3.3.4.3. Select the type of device you're adding. In this case, click Xbox 360.

1.a.3.3.4.4. Enter a name for this Xbox 360 console, and click Next.



1.a.3.3.4.4.1. This name serves only to identify your Xbox 360 console to XNA Game Studio. The name does not need to correspond to any other computer or Xbox 360 name.

1.a.3.3.4.5. Enter the connection key that is displayed in XNA Game Studio Connect on the Xbox 360.



1.a.3.3.4.5.1. The connection key might be somewhat hard to read on a standard television screen. The following guide should help you identify specific letters and numbers:

1.a.3.3.4.5.1.1. The number "1" has a small tick at its top left; the capital letter "I" does not.

1.a.3.3.4.5.1.2. The capital letter "B" has a straight line on the left; the number "8" does not.

1.a.3.3.4.5.1.3. The number "3," sometimes mistaken for a "B," also has no straight side on the left.

1.a.3.3.4.5.1.4. The number "0" and the capital letter "O" are so similar that XNA Game Studio Connect treats these characters as the same. Therefore, the number "0" and the capital letter "O" are interchangeable.

1.a.3.3.4.5.1.5. If the connection key is still too difficult to read, press X on the Xbox 360 controller to generate a new connection key.

1.a.3.3.4.6. Once you are sure that the two keys match, click Next on the XNA Game Studio Devices dialog box.

1.a.3.3.4.7. XNA Game Studio Device Center will test the connection with the Xbox 360 console.

1.a.3.3.4.8. If the connection is successful, the XNA Game Studio Device Center on the Windows-based computer will display "Successfully connected to the Xbox 360 console." XNA Game Studio Connect on the Xbox 360 console will display "Waiting for computer connection," followed by the name you have chosen for your Xbox 360 console in the XNA Game Studio Device Center.

1.a.3.3.4.9. If the XNA Game Studio Device Center fails to connect to the Xbox 360 console, click Try again to edit the connection key and try again. If the connection continues to fail, make a careful note of the error message displayed at the bottom of the XNA Game Studio Devices dialog box. This error message can help you or a technician diagnose the cause of the connection failure, if it did not result from mismatched keys.

1.a.3.3.4.10. Click Finish.

1.a.3.3.4.10.1. The name you gave to your Xbox 360 console will be listed in the XNA Game Studio Device Center. From now on, your computer and your console can connect to each other easily.

1.a.3.4. Step 4: Create and Deploy an Xbox 360 Project

1.a.3.4.1. Try out the Xbox 360 console by deploying a simple, blank XNA Game Studio game to it.

1.a.3.4.2. On your Windows-based computer, at the main Visual Studio screen, select the File menu, and then click New Project.

1.a.3.4.3. From Project types, expand the Visual C#, and click XNA Game Studio 3.1.

1.a.3.4.4. In Templates, select Xbox 360 Game (3.1), and then click OK.

1.a.3.4.5. At this point, you should bring up XNA Game Studio Connect to prepare the Xbox 360 console to receive content from the Windows-based computer.

1.a.3.4.6. From the Xbox Dashboard, go to My Xbox, select Game Library, and press A on the controller.

1.a.3.4.7. You can also use the Guide: select the Games tab, select Game Library, and then press A.

1.a.3.4.8. From the Game Library, go to the Collections tab, select Community Games, and press A.

1.a.3.4.9. Select XNA Game Studio Connect, and press A.

1.a.3.4.10. Select Launch, and press A.

1.a.3.4.11. The XNA Game Studio Connect screen appears.

1.a.3.4.12. On your Windows-based computer, with your new project open, press the F5 key.

1.a.3.4.12.1. The project will build, deploy necessary files to the Xbox 360 console, and run.

1.a.3.4.12.2. At this point, you should see a simple display on your console—just a blue screen. If you see this screen, you have

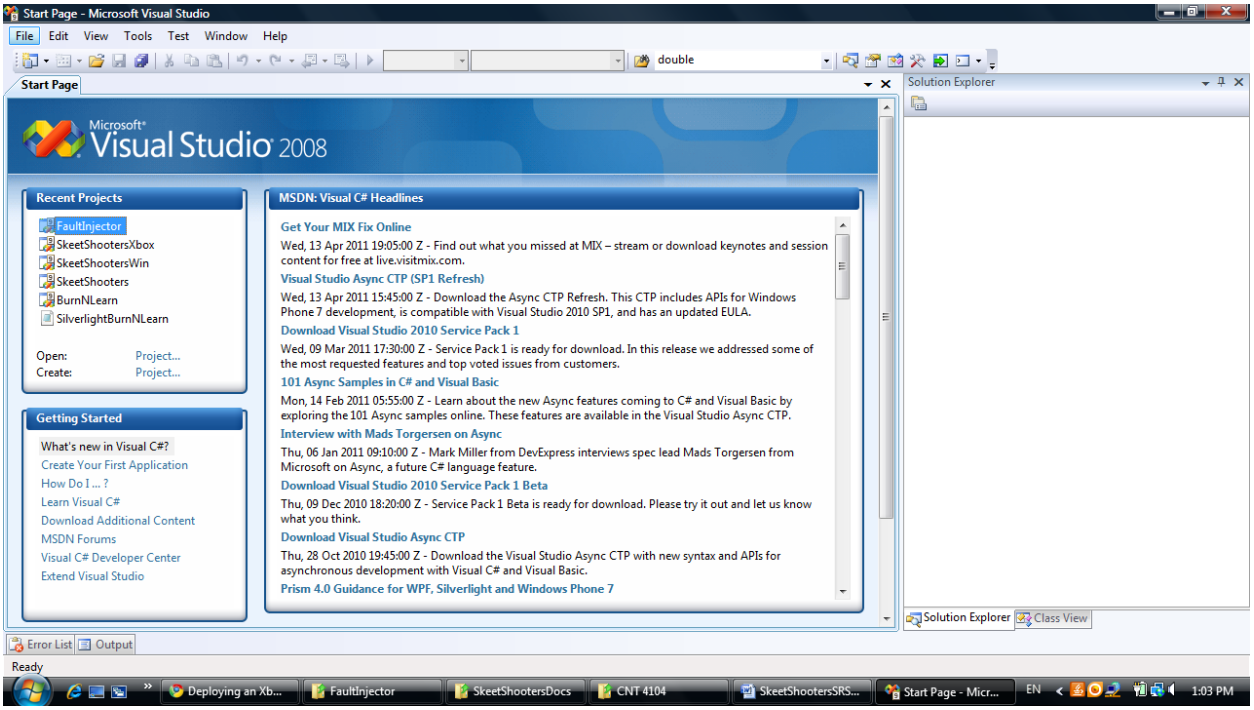
successfully deployed a game to your Xbox 360 console. You can now deploy and play any game you create on your console.

1.a.3.4.13. To stop the game and return to the main screen of XNA Game Studio Connect, either press the BACK button on your Xbox 360 gamepad, or press SHIFT+F5 to stop debugging on your computer.

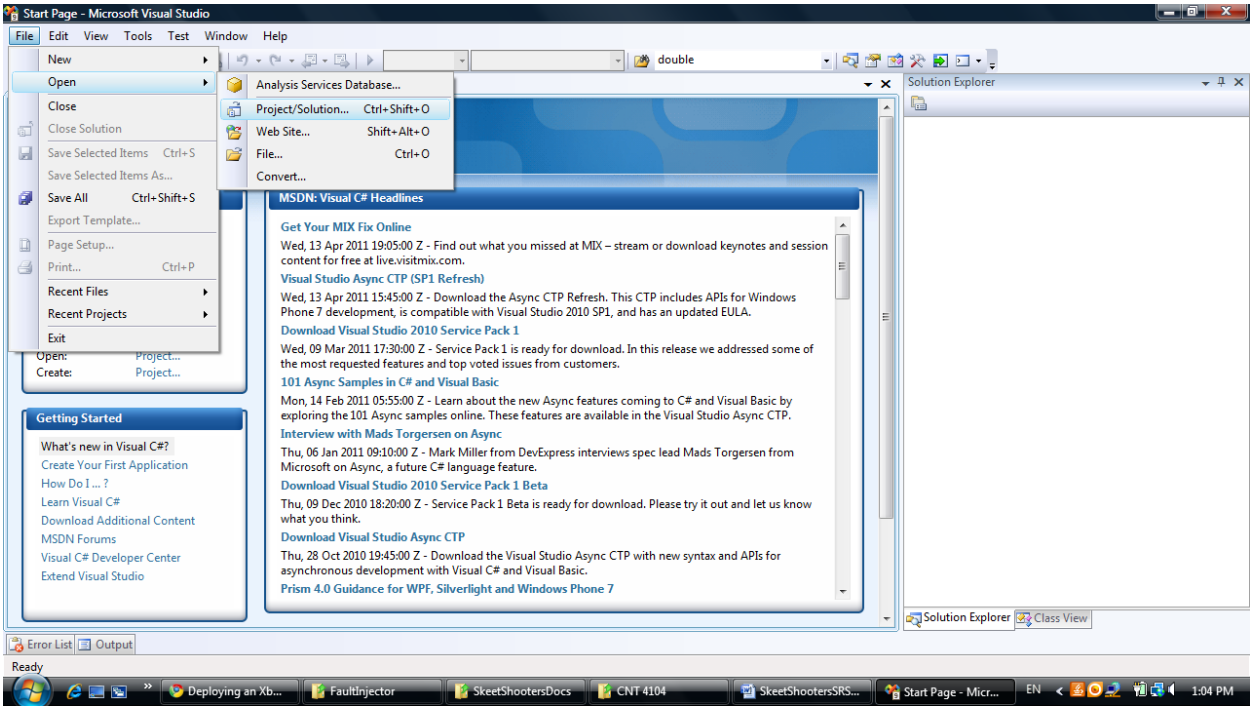
4. Opening an Existing XNA Project

1.a.4.1. Run Microsoft Visual Studio 2008.

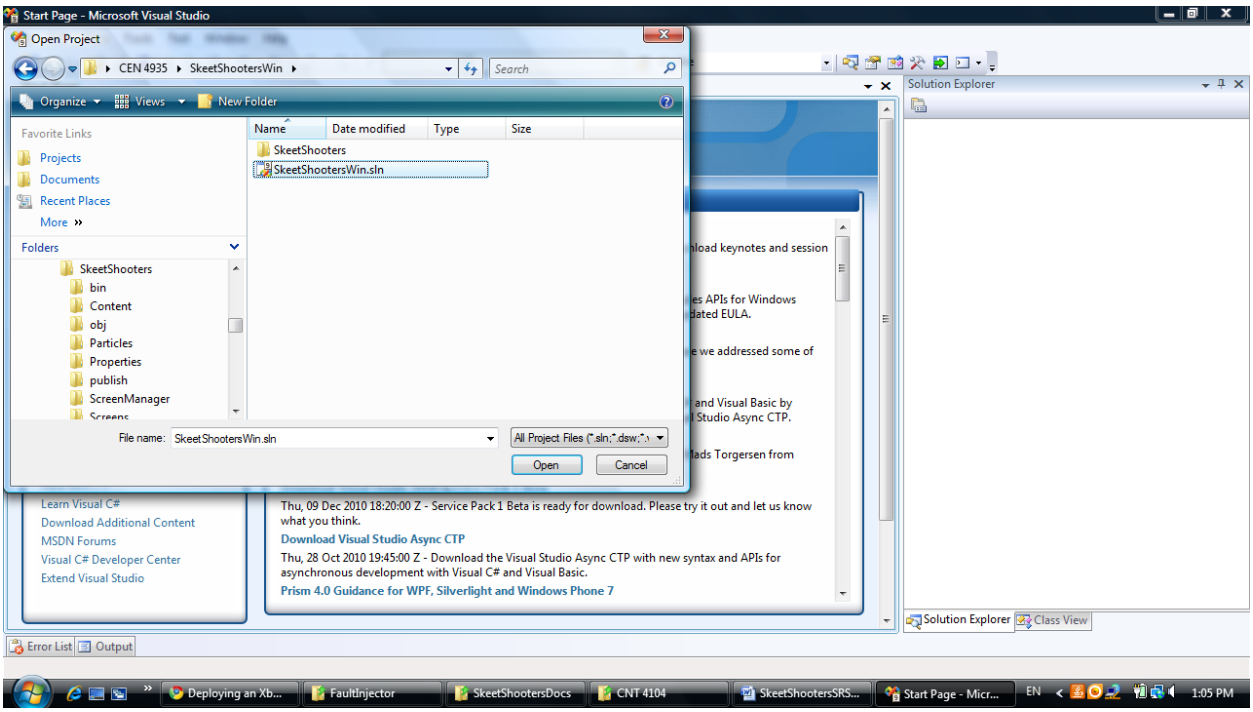
1.a.4.2. Click on the File drop-down menu.



1.a.4.3. Click Open and then Click Project/Solution

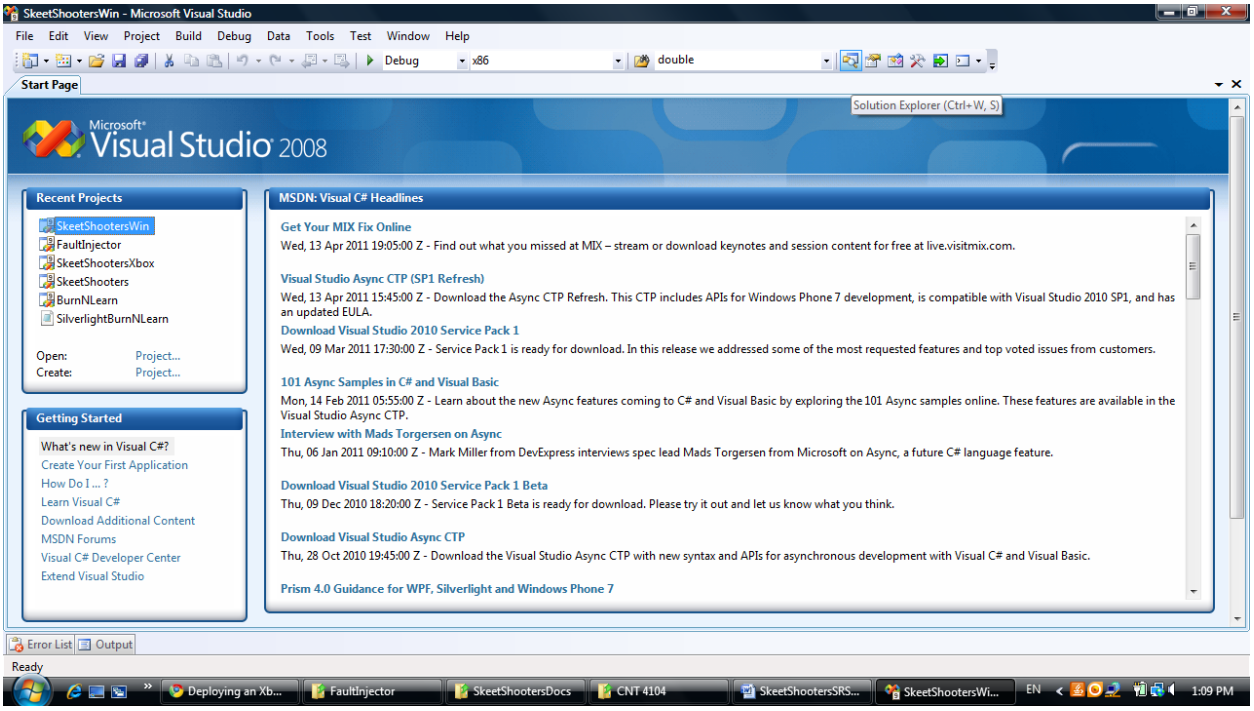


1.a.4.4. Locate the folder containing the XNA project solution.

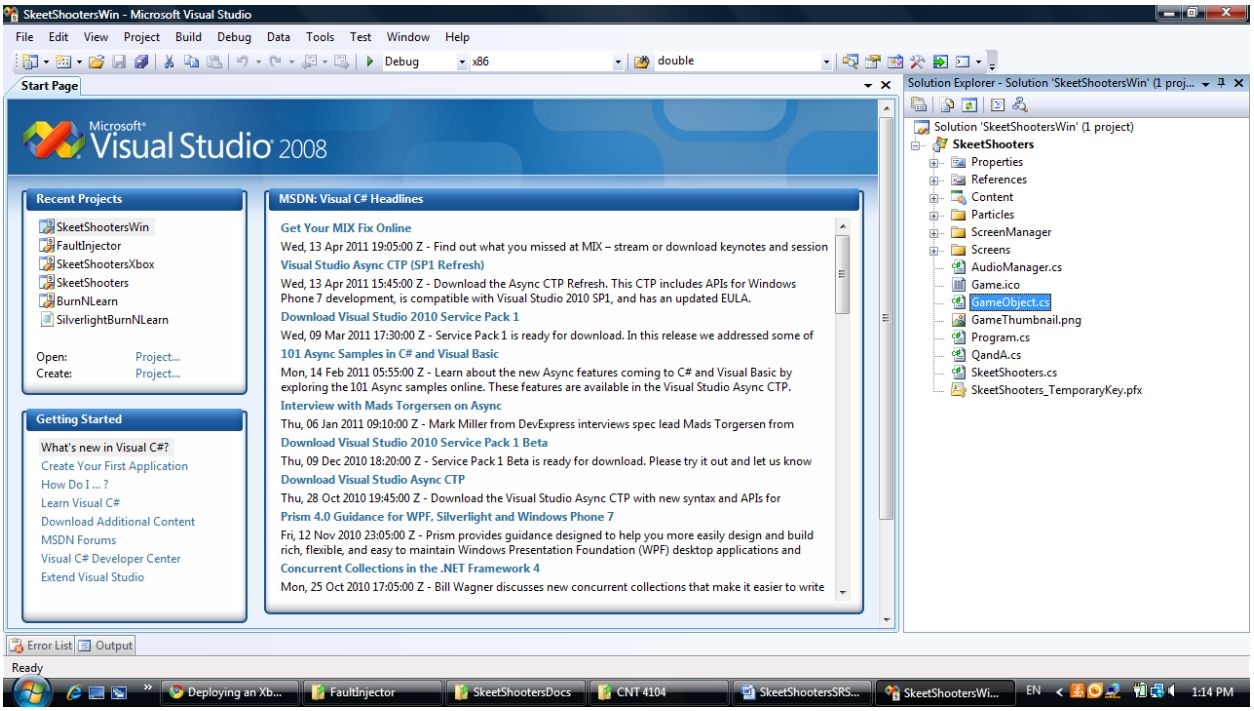


1.a.4.5. Click on the solution and then Click Open.

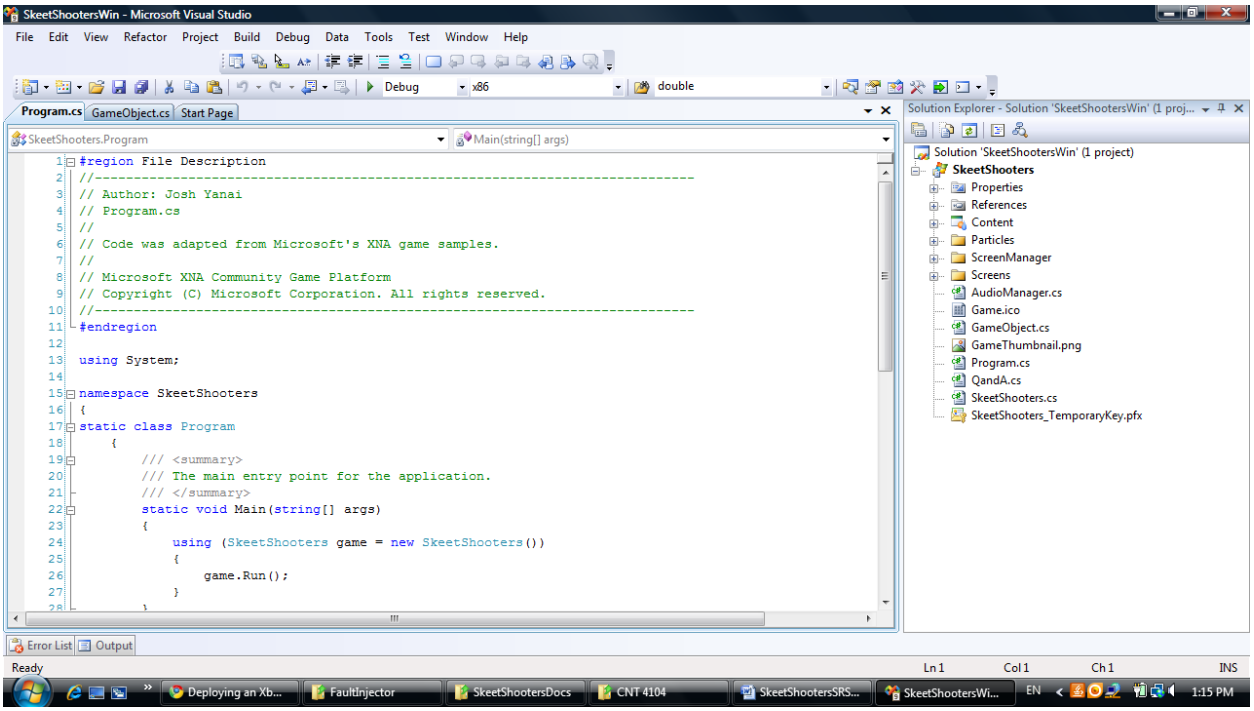
1.a.4.6. Open the solution explorer by clicking the solution explorer icon



1.a.4.7. The solution explorer contains all of the game's classes, sounds, music, fonts and images.



1.a.4.8. Double clicking on one of the C# class icons, in the solution explorer, will open the respective class's code.



1.a.4.9. Pressing F5 will build and run the project, assuming everything has been set up correctly.

5. Playing the Skeet Shooters Game

Xbox 360 Controller Layout

1.a.5.1. Menu Screens

1.a.5.1.1. Every menu screen contains a list of menu entries

1.a.5.1.2. Use the Left Stick or the Directional Pad to highlight a menu entry

1.a.5.1.3. Press the A button to select the highlighted menu entry.

1.a.5.1.4. Main Menu Screen

1.a.5.1.4.1. Selecting “Start Game” displays the Skeeter Selection Screen where the user can select their character.

1.a.5.1.4.2. Selecting “Options” displays the Options Menu Screen

1.a.5.1.4.3. Selecting “Exit” will terminate the program.

1.a.5.1.5. Skeet Selection Screen

1.a.5.1.5.1. Selecting a character displays the Game Play Screen and begins a new game with the selected character.

1.a.5.1.6. Game Play Screen

1.a.5.1.6.1. Use the Left Stick or Directional Pad to move your character (a.k.a Skeeter).

1.a.5.1.6.2. Pressing the A button shoots a bullet.

1.a.5.1.6.3. Pressing the X button enables your shield.

1.a.5.1.6.4. Pressing the Start button displays the Pause Menu Screen.

1.a.5.1.7. Pause Menu Screen

1.a.5.1.7.1. Selecting “Resume Game” resumes the current game.

1.a.5.1.7.2. Selecting “Sound” displays the Sound Menu Screen.

1.a.5.1.7.3. Selecting “Quit Game” exits the current game and displays the Main Menu Screen.

1.a.5.1.8. Options Menu Screen

1.a.5.1.8.1. Selecting “Difficulty” changes the difficulty level.

1.a.5.1.8.2. Selecting “Tutorial” displays the Game Tutorial Screen.

1.a.5.1.8.3. Selecting “Sound” displays the Sound Menu Screen.

1.a.5.1.8.4. Selecting “Back” displays the Main Menu Screen.

1.a.5.1.9. Sound Menu Screen

1.a.5.1.9.1. Selecting “Music Volume” displays the Music Volume Screen.

1.a.5.1.9.2. Selecting “SFX Volume” displays the Sound Effects Volume Screen.

1.a.5.1.9.3. Selecting “Next Song” stops the currently playing song and plays the next song in the music queue.

1.a.5.1.9.4. Selecting “Prev Song” stops the currently playing song and plays the previous song in the music queue.

1.a.5.1.9.5. Selecting “Back” displays the Options Menu Screen.

1.a.5.1.9.6. Selecting “Next Song” stops the currently playing song and plays the next song in the music queue.

1.a.5.1.10. Music Volume Screen

1.a.5.1.10.1. The top menu entry displays the current music volume level.

1.a.5.1.10.2. Selecting “Volume Up” increases the current music volume level by 10%.

1.a.5.1.10.3. Selecting “Volume Down” decreases the current music volume level by 10%

1.a.5.1.10.4. Selecting “Back” displays the Sound Menu Screen.

1.a.5.1.11. SFX Volume Screen

1.a.5.1.11.1. The top menu entry displays the current sound effects volume level.

1.a.5.1.11.2. Selecting “Volume Up” increases the current sound effects volume level by 10%.

1.a.5.1.11.3. Selecting “Volume Down” decreases the current sound effects volume level by 10%

1.a.5.1.11.4. Selecting “Back” displays the Sound Menu Screen.

1.a.5.1.12. Game Tutorial Screen

1.a.5.1.12.1. Pressing the Start button displays the Pause Menu Screen.