Space Invaders: Mark I & II

Overview

Recreate a classic 2D arcade game using the SDL to support graphics, input and audio abilities as well as developing an understanding of game engine models and features appropriate for this type of classic arcade game.

The specific topics to explore in this work includes:

- Identification and experience with game engine features required for games of this genre
- The creation of object oriented models for sprites and other game related entities
- Using a simple media library (SDL2) for sound, images and input handling
- Simple 2D box collision models as well as low-level pixel based collision models
- Event-based sound and background music playback
- Keyboard input state (key-press-up/down) handling and input mapping management
- Finite state machine (FSM) models to represent game entity behaviour and associated properties
- Use and comparison of bitmap fonts and true-type fonts
- The creation and use of animated sprites (as an extension of a simpler sprite models
- Design and development of "game option" managers for game input (key mapping), sound and music
- Appropriate OO game state (stage) management

Space Invaders is a classic 2D arcade game created by Tomohiro Nishikado in 1978. It is one of the first shooting games. Although very simple by today's standards, when first released in Japan as a coin-op arcade game it was very popular and created a temporary shortage of 100-yen coins. It is later credited with quadrupling the sales of the Atari 2600 making it a "killer app" of the time.
> http://en.wikipedia.org/wiki/Space_Invaders for the usual wikipedia goodness
> http://www.spaceinvaders.de/ to play this classic online
> http://www.klov.com/game_detail.php?game_id=9662 for the klov game details

Game Play Screen

To discuss this game in more detail a list of game tokens is useful. Tokens match easily to game objects, but do not need to be implemented as simply one object per token. See Figure 1 for a mockup of the in-game screen and the tokens identified.

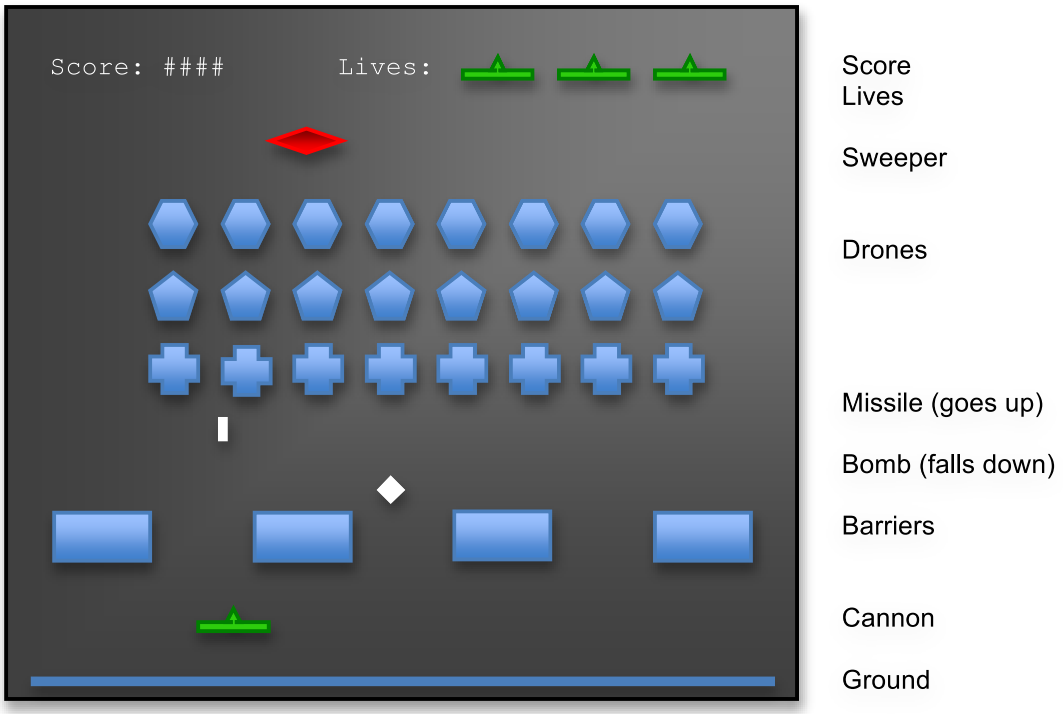


Figure 1. A basic "mock up" style representation of the game screen and names used to identify the game tokens (objects) discussed in the interaction section

Game Tokens and Interactions

From Figure 1 we can now create Figure 2 which lists game “tokens” (or “objects” if you wish) and their interaction in the game. You may decide to create different tokens in your specific implementation. You should implement your game so that only the interactions required are tested – don’t waste time on tests and operations that aren’t needed!

	Missile	Bomb	Cannon	Drone1	Drone2	…	Sweeper	Ground	Barrier	Score	Lives
Missile (up)	#										
Bomb (down)	-	#									
Cannon	-	HIT *Death*	#								
Drone I	HIT+	-	-	#							
Drone II	HIT+	-	-	-	#						
…	HIT+	-	-	-	-	#					
Sweeper	HIT+	-	-	-	-	-	#				
Ground	-	HIT	-	-	-	-	-	#			
Barrier	HIT *Damage*	HIT *Damage*	-	-	-	-	-	-	#		
Score	-	-	-	-	-	-	-	-	-	#	
Lives	-	-	-	-	-	-	-	-	-	-	#

Figure 2. Token interaction matrix for the Space Invaders game. The “…” is used as a general description for all other alien invading “drone” ships not listed. Note that “#” is used for self-self interactions, and “-" to indicate no interaction. HIT indicates collision detection, and HIT+ indicates that the player score increase. The Bomb-Cannon collision results in the player’s death, and Missile-Barrier or Bomb-Barrier collision damages the Barrier until it is gone.

Game States

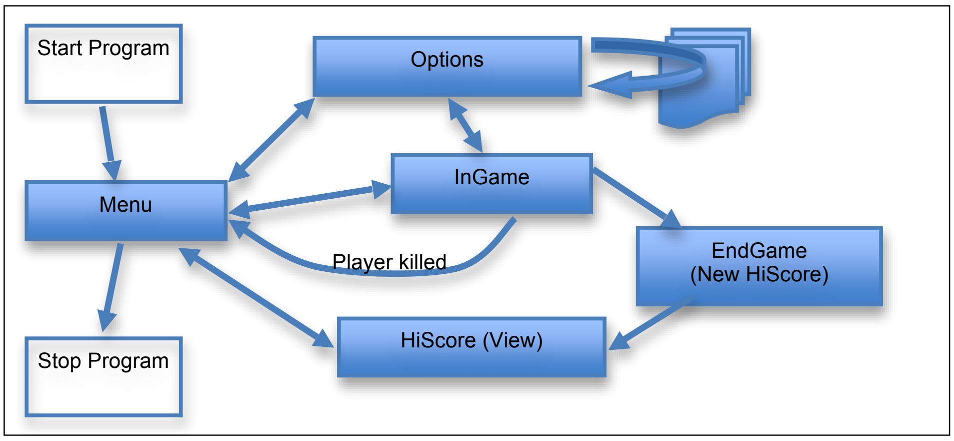
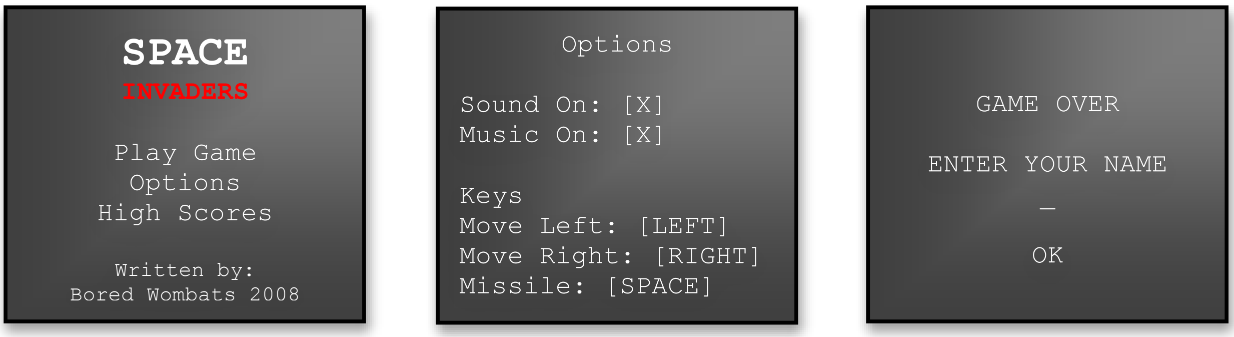


Figure 3. A state map to represent the different game “stages” required in Space Invaders. Note that the Options state may contain additional sub-states but it is not required to be done in this way.

Game Screens



SPACE
INVADERS

Play Game
Options
High Scores

Written by:
Bored Wombats 2008

Options

Sound On: [X]
Music On: [X]

Keys
Move Left: [LEFT]
Move Right: [RIGHT]
Missile: [SPACE]

GAME OVER

ENTER YOUR NAME

_

OK

Figure 4. Three basic screen representations for the game “Menu”, “Options” and “Game Over” screens.

Game Features and Techniques

Table 1 lists the main game programming techniques required for Mark I and Mark II versions of the space invaders game. Some of the concepts are presented in more detail later in this document.

Table 1. Game programming techniques required for the different game versions.

Technique	Mark I Requirements	Mark II Requirements
Sprites	• Simple single image • Position and basic state • Box-model collisions	• Animated (multiple frames) • State managed by FSM • Pixel-based collision
Input (keyboard)	• Simple key-state	• +key mapping (stored as game data)
Sound & Music	• Shooting, explosions, background	• *(no change)*
Font	• *(None)*	• Bitmap and True-Type
Game Stages	• InGame only	• Menu, Options, InGame, EndGame, HiScore
Player State	• Simple Alive-Dead	• FMS:
Game Data	• Single level • Player Alive/Dead • *(No lives/score)* • *(No options manager)*	• Multiple levels with difficulty changes • Player Lives (3) • Player Score (kills + bonuses) • Game Options (sound, music, keys)

Space Invaders Mark I consist only of the game play (InGame) state and nothing else. It also does not contain player “lives” or any scoring. The focus of Mark I can be listed as the following game features

- graphical presentation of game tokens (sprites and code making use of sprites),
- processing of user input (key-state) to move the player Cannon and to shoot Missiles,
- updating of a game world model that includes
  - invading hordes of alien drone ships,
  - occasional appearance of the Sweeper ship and its movement across the screen
  - dropping of alien Bombs,
  - collision detection using simple box model technique
- playing of background music and event sounds during the game.

The destructible barriers (which can protect the player’s Cannon) are optional.

Space Invaders Mark II is a complete game, including game states and additional presentation and game-play features. It must include the following

- animated alien Drones (alternation between different frames)
- animated Cannon death sequence
- the inclusion of player Lives and Score as game data
- the addition of finite state machines for game entities
  - simple alive-dead state for basic drones
  - complex multi-hit (health) for harder drones
  - player Cannon “new”-“alive”-“dying” states
- game state (stage) management including “game over” state and game “option” management
- game option management that allows adjustment of music, sound and keyboard settings
  - sound options can be simple on/off or more advanced “level” control
  - key-mapping is a central objective (as a general player input/control mapping concept)
- updating of the world model that includes
  - collision detection using pixel based techniques
  - game level management (new-horde and horde-destroyed etc)
- Bitmap fonts for text and numbers in the game
- True-Type fonts for options and high-score screens.

See Table 1 presented earlier for a summary of the technique differences between Mark I and Mark II of the game.

Sample Screens

Figure 5 contains screen shots of an online recreation of the classic game. It is a good reference for game play. However note that the menu requires are different, and that the mark "I" version of this specification is a lot simpler that the game in the example.

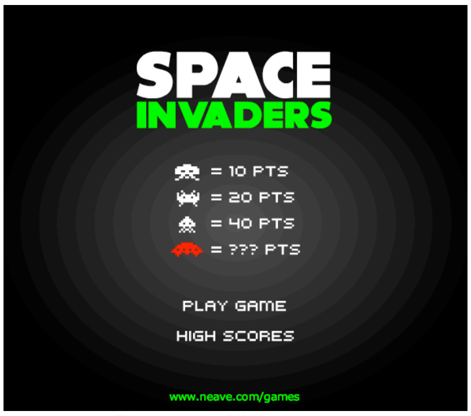


Figure 5a. Menu stage

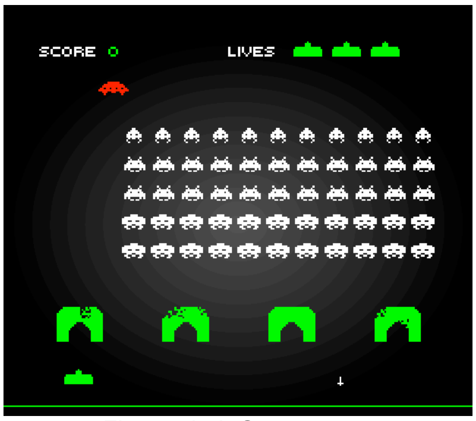
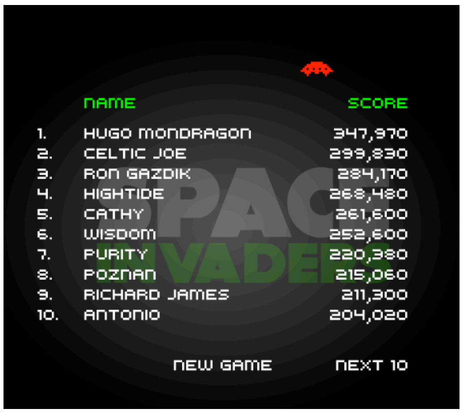


Figure 5b. InGame stage



	NAME	SCORE
1.	HUGO MONDRAGON	347,970
2.	CELTIC JOE	299,830
3.	RON GAZDIK	284,170
4.	HIGHTIDE	268,480
5.	CATHY	261,600
6.	WISDOM	252,600
7.	PURITY	220,380
8.	POZNAN	215,060
9.	RICHARD JAMES	211,300
10.	ANTONIO	204,020

Figure 5c. HiScore (view) stage

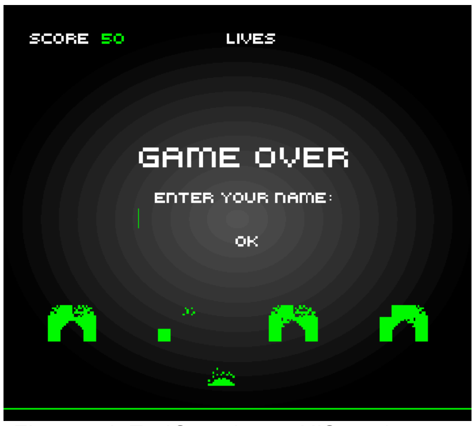


Figure 5d. EndOver (enter HiScore) stage

Figure 5(a-d). Screen shots of Space Invaders as visual examples taken from
http://www.neave.com/games/spaceinvaders/spaceinvaders_external.swf