JAWS *Language*

Justin Lu | jl1439@columbia.edu [leader]
Andy Guo Xin | gl350@columbia.edu
Winston Chao | wc386@columbia.edu
Shoaib Anwar | sa597@columbia.edu

1.1 Introduction

Anyone who grew up in the mid-eighties to early-nineties is probably familiar with the genre of video games known as space shooters. While playing titles such as Galaxian and Galaga, most of us could not help but wonder how cool it would be if we were able to create our own space shooter type game without having to know complex graphics programming.

The JAWS language is the culmination of that dream, a simple programming language designed to allow users to easily create a vast array of space shooter type games. One of the main goals of JAWS was to allow for the creation of space shooter games without needing knowledge of graphics programming. The purpose of JAWS is to free game designers from that burden so that they can focus on the design of gameplay and depth in space shooter games.

1.2 JAWS - Main Features

Simplicity

The JAWS language is designed so that casual programmers or non-programmers with a good grasp of logic can be able to make space shooters. If the language wasn't designed for ease of use, then it would be obsolete since a space shooter could just be programmed from the ground up. Why JAWS is useful for the casual programmer is because it uses modern, easy to understand syntax. The structure of the language is designed to mimic most languages of today such as JAVA and C++. This will eliminate the need to learn a whole new set of rules for the semantics of the language.

Ease of Use

JAWS totally eliminates the need of any knowledge of graphics programming by incorporating most of the techniques used right into the language's design. The one biggest hump to any programmer when wanting to program a game is having to program the graphics. JAWS allows its user to bypass this by having most of the necessary tools in space shooter graphics, logic, and physics built-in as predefined functions.

For one, collision detection will be integrated into JAWS' object oriented nature. Objects such as ships, enemies, will contain flags or methods which allow to the user to access distance from each other or whether two objects are colliding. The collision detection only serves to inform the user how far two objects are from each other or whether they are colliding, it is up to the user to take that information and decide how they want their game to react to it. JAWS was designed this way so as to not make the gameplay too preset.

Game physics will also be incorporated as part of the language so the user does not have to deal with that aspect in creating a space shooter. Movement patterns of objects on the screen can be specified in the syntax. For example, enemies can be designated to move in circles, zig-zags, etc.

Speed and acceleration of enemies or the player can also be easily assigned. However, JAWS will also allow for the advanced user to specify custom game physics if he or she so wishes.

Object-Oriented

JAWS is designed as an object-oriented language to further extend the idea of simplicity of the language. Object orientation facilitates the re-use of code which can be extremely desirable in creating a space shooter game. For example, entire levels can be generalized by specifying general attributes such as number of enemies, locations, and etc. but can be re-used to make multiple levels by changing the individual AI or difficulty. Each item in JAWS will be designed to incorporate object-orientation.

Portability

Since JAWS will be implemented using JAVA, it automatically inherits the portability of JAVA. Space shooters will be able to be generated on any platform and played on any other platform. This helps users who wish for ease of distribution of their space shooter games.

Customizable

Even though JAWS provides default graphics, AI, movement patterns to speed up the process of putting together a space shooter, the language gives the user the flexibility to fully customize almost every feature. The player's ship, enemies, and surroundings can be changed to images that the user can specify. Furthermore, predefined AI for enemies can be reprogrammed by the user if the user is interested in making AI especially easy or hard. With the power of a customizable language, the space shooters that are created don't need to all look the same.

Built-in Sound Support

A very important feature of any game is sound. Sound effects and background music help to enhance the atmosphere of a game. JAWS will have built-in support to allow users to specify .midi background music, as well as have default sound effects for collisions, movement, and etc. Of course, the sound effects will be easily customized if the user so desires.

1.3 Syntax Sample:

Declaring an enemy object:

```
Enemy(name:badguy type:1 hp:100 movement:circle);
```

Detecting Collisions:

```
//where badguy is the name of an enemy
//player is the name of the player's ship
//~ is the distance operator


if (badguy~player == 0)
{
   label1.display("YOU DIED");
   player.lives -= 1;
}
```