Assembly Airplane's Astonishing Acceleratory Advancement And Annihilation

A simple game about a tiny paper plane, a distant yellow planet, spikes, death, and nothingness
by Igor (Ihor) Antonov

Introduction

What follows is a document describing various peculilarities of a **three-dimensional** spike-dodging and aircraft-maneuvering game program written in 100% Motorola 68000 assembly called Assembly Airplane's Astonishing Acceleratory Advancement And Annihilation (AAAAAAA for short, or simply *the game*). The game is about a little airplane which flies towards a distant star evading dangerous spikes. AAAAAAA was a great learning experience in regards to the capabilities and limitations of 68000 assembly programming and particularly the EASy68k simulator.

Premise

For we brought nothing into this world, and it is certain we can carry nothing out.
1 Timothy 7

The game tells a story of a little, yellow, paper airplane, whose fickle life hangs on a thread facing the threat of devious blue pyramid-spikes. It is up to the player to let the airplane survive by utilizing their input input device to affect the aircraft's trajectory and steer it away from a certain demise.

The airplane **starts with nothing**, but the player's benignantly aidful disposition, in a dark, cold, uncaring world full of terrible pointy hazards; bright circle, similar to that of the sun, glaring in the distance, bright yet giving no heat to that lifeless expanse. The sky is dark, enhancing the feeling of emptiness and **nothingness**, seemingly due to the insufficient density of the atmosphere.

Left and right it slides, in an unending dance of life and death, going faster and faster, each level seemingly flying with increased rapidity (just like the days of your life with age), and then, if or when (it is a question of time) the player makes an unavoidable mistake, the paper plane crashes. And dies. With **nothing**.

Gameplay

You control the tiny yellow airplane with your keyboard or controller (if set up) and try to avoid the spikes while enjoying the sun and the landcape.

Controls

Key	Function
A	Left
D	Right
Space	Confirm
F	Toggle Fullscreen

Development Process

Originally developed as a simple assembly experiment, which was supposed to test out the possibilities of 3d point projection on a 2d viewport plane, the game slowly grew into a maze game with a tilemap and item interaction. A week before the contest the original game was scrapped due to supposed complexity and some clipping issues, and was replaced with the airplane game we have now.

Overall Code Organization

The game makes liberal use of bsr and rts assembly instruction to facilitate abstraction, code reuse and clarity of intent when it comes to the purpose of the code. Various builtin traps of the simulator were wrapped in subroutines for ease of use. Most of the subroutines have comments with the list of arguments and which registers they go into.

Wireframe Graphics

The core part of the game which gives the game it's specific look.

The following formula is used to project the point across each of the coordinate axes:

$$x_{projected} = d\frac{\Delta x}{\Delta z}$$

Where d is the distance between the player and the viewport, Δx is the difference between player x and the x of the point being projected, and Δz is the difference of the z of the player and the point being projected.

The same calculation can be performed for the other coordinate of the projected point.

Collision Detection

Collision detection is implemented by the simple means of rounding the coordinates of an object and comparing them to the value of the corresponding tile on the map.

Art and Design Choices

Wireframe graphics were chosen due to a certain simple, yet beautiful quality to them, eliciting the look and feel of old vector displays and arcade machines, the PLATO system, BBC Micro Elite and Battlezone Arcade.

Background Graphics

The background graphics are drawn using basic geometric primitives supported by EASy68k, such as ellipses and rectangles.

Music

The music was made by a good friend of mine who desired to stay anonymous due to videogame music not being the kind of music he actually enjoys making most of the time and the lack of desire to associate with gamedev. If you really liked his music and want more I can ask him if he wants to contact you.

Model and Character Design

I would like to thank the friends who helped me to design a more recognizable paper airplane model by providing feedback and suggestions.

The Plane

The plane was chosen due to it's beauty, familiarity and simpliciy. Everybody made a paper plane in their life. Just by folding a piece of paper a couple of times one can get an improvised airborne machine. It's aerodynamic shape is the very embodyment of the idea of going forward.

The Pyramids

Pyramids carry their own deep symbolism, from the pointy shape which clearly signifies danger to the player, to their cultural significance around the world and them being close to the natural shape of the structure which forms when particles or things in general fall haphazardly on top of each other in the so-called 'pile'.

Lessons Learned

When programming in assembly, great attention to detail is required, both to the operation of your program and the workings of the machine itself, namely, the little details of each and every command that your program might make use of throughout its execution.

During the development of this project I truly felt how much is the cognitive load is laid upon the brain of the programmer compared to a higher level language like C where, thanks to the language's more advanced means of abstraction, less cognitive effort is required to express a more complex idea, which leads to a lower bug density (bug quantity / idea complexity). Thus it is good to use assembly when the benefit gained by using it outweights the loss of programmer time.

Also, it is important to define project boundaries based on the time constraints and project difficulty.