YES, THIS IS

EMBARASSING. AND

YES I MARKED OUT A

FEW NAMES TO PROTECT

THE GUILTY. ☺

2/27/2017

SUPER SPACE ACE R

JAGUAR PACK 1995

SSA

MIDBOSS

SPACE SATELITE
HAS MIRRORS
THAT REFLECT
A SIMULATED
STARFIELD

SSA 2

ROLL TEXT

USES

@ ' FOR APOSTROPHE

END WITH

(DID YOU NOTICE
WE USED REAL APOSTROPHES
WITH CURLY TAILS?)

IF CAMERA WAVES AT
COCKPIT AND SAYS
"HI MOM!"

SSA - IDEAS

SHOOT MOVING TARGET
MAN - ~~BUTTON~~

END - END OF
PROGRAMMING DAY
SCROLL ANTHEM
TEST PATTERN

GORD'S AI
- RANDOMLY STOPS,
PANS LEFT / RIGHT,
THEN TAKES OFF
AFTER NEW TARGET
(NOT EVERY TIME)

TEXTURE
MAP EVERY
2ND PIXEL

- CAN THEN
LIGHT SOURCE
BY CHANGING
COLOUR OF
UNDERLYING
POLYGONS

ANALOG CONTROL?
IN-COCKPIT VIEW

KONAMI 3D-MAN CODE
WORKS

~~HOLD FOR START~~
~~COCKPIT FLOW~~

SSA
LIGHTING
&
SHADING
? IMPORTANT !

STAPLES

1 IN 150 CHANCE ON BOOT THAT TITLE PAGE/ MUSIC FOR 'THE ADVENTURES OF FLUFFY THE BEAR & FRIENDS' COMES UP (OR HOLD 3 BUTTONS)
- ON BUTTON PRESS "(OK, JUST KIDDING)" (START NORMAL INTRO)

MORPHING SCORE CHANGE

HITCHHIKERS -> O MORPH

O SOLITAIRE

SUPER SLEEP ACER

DIFFERENT DEATH EFFECTS
- DISINTEGRATE, AMBULANCE TRANSPORTER

SSA

ENEMIES/SHOTS/ OBSTACLES/ ETC *NOT* LINED UP ON THE Y-AXIS ARE 'MASKED-OUT' (LIKE NON-SELECTABLE MENU OPTIONS OFTEN ARE) (DIFFERENT COLOUR FOR ABOVE AND BELOW?)

SSA
DEBRIS IN SPACE - SOME BEING AN ECM MAGAZINE.

EAGLE HIDDEN UNDER SIDE OF WINGS

LION ICON FOR GREAT SPEED

SSA

- 2nd JAGUAR NETWORK SUPPORT FOR VIEWING ACA VIRTURACING EXTRA MONITOR
- FLAME TRAILS ON ~~ENGINES~~ SSA
- RAY CASTING - LIGHTING
- ROTATING VECTOR BOSS VIEWS
- TEXTURE MAPPED POLYGONS.

- MULTIPLE VIEWPOINTS OF BOSS DESTRUCTION WITH CAMERA PANNING
- USE ROTATING SHIP VIEW IN ATTRACT MODE, DISPLAY STATS A LA ST VERSION (REMEMBER 'ECONOMY MODE')

SILLY PAGE

- MORTAL WOMBAT ON 1ST BOOT ✓ IN
- USE SIGHT TO CHASE STICK MAN
- LLAMA SPACE INVADERS
- ASTEROIDS WITH SSA SHIP (WITH PROGRAMMER HEADS AS ROCKS)
- PACMAN CHASING SHIP?

- PROGRAMMERS HEARS AS HIDDEN BOSSES
- SCHOOL BUSES
- STARS BECOMING CHICKENS
- BOOM MIKES LOWERED DURING CINEMAS
- HAPPY FACES IN Ø'S (STICKING OUT TOUNGES)
- 32 BIT SCORE
- PEOPLE WHO REFUSED TO APPEAR IN THIS GAME
- SCREEN: "JAGUAR: EVEN BETTER THAN THE R.E.A.L. THIN
- SPEED: AVERAGE
  WEAPONS: ECONOMY MODEL
- "RAYSTATION" - ENEMY BOSS = SUNNY RAYSTATION

CODE TO MAKE A BOSS RUN IN FEAR
(BIG-EYES, YELPING)


SUPER
SPACE
ACER
(C) Moving Target
Software Design


89%
SCORE
0000013

Non-Disclosure Agreement

All subject matter, ideas, discussion, mention of our existance, names, dates, really secret stuff, personal vendetta's, and so forth with regard to **Moving Target Software Design** projects are simply not allowed to be mentioned by any Atari employee or agent outside of this room. Not that we are stuck up, but this osmosis theory we are operating on is bad enough for spreading our ideas, we don't need to lose anymore.

And be sure to have a great E^3!

_____

Gordon Haddrell
Supere Dicator
Moving Target

_____

Mike Brent
Minion
Moving Target

_____

Bill Rehbock
Vice President
Software Business Development
Atari Corporation

_____

Anyone Else
Atari Corporation

Date: _____

Super Space Acer

The Plan

(Boring legal stuff)

Super Space Acer (SSA), it's design, and all associated intellectual properties are owned by Mike Brent, licensed to Moving Target Software Design. The following information is provided in confidence solely for the purpose of Atari's feasibility decision, in order to obtain a Jaguar development system. All information and ideas are protected by the Non-Disclosure Agreement (NDA) between Atari and Moving Target.

Table of Contents

What is Super Space Acer?

Super Space Acer is, in it's simplest form, a 3d-style outer space shooter, intended as the first Jaguar project for Moving Target Software Designs (pending approval from those great guys at Atari, of course).

Super Space Acer chronicals the trials of a janitor by the name of Bob Shumway. Bob is obsessed with the idea of becoming a fighter pilot, to the point where his distraction on the job becomes dangerous to others on the starship he works on.

Too polite and civilized to merely fire or remove him, the ship's admiral comes up with a 'feel-good' way to get rid of him, and to aid in their war against the evil Qwertians at the same time. Bob is sent off in a rickety old, barely serviceable fighter named the 'Snowball', straight into Qwertian front lines, where it is hoped he will provide adequate diversion for the Union to mount a major offensive.

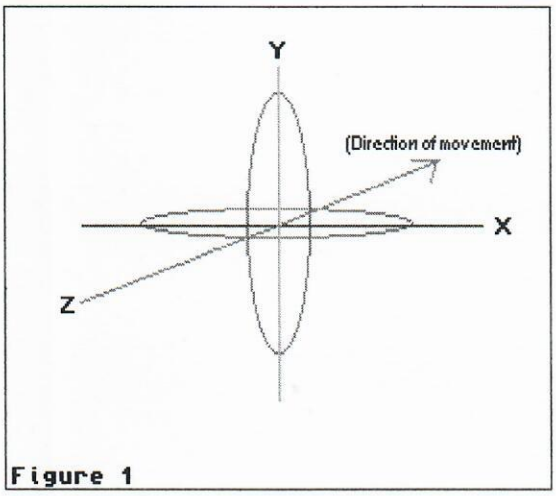

Figure 1

The game itself takes up where the story leaves off, just as Bob begins to face resistance. The game employs a 3d-perspective, the angle of which is user-controllable. The ship itself is controllable on the X-Y plane, while moving constantly forward on the Z axis (assuming X to be left-right, Y to be up-down, and Z forward-back, see fig 1.). The speed of forward movement may also be controlled. The view angle is dynamic, and may be altered by the user. The 'camera' resides on a plane behind the ship, aiming forward in the ship's direction of travel, while also showing the ship. The camera may be located at any of 9 locations at the choice of the player, corresponding roughly to the locations of the buttons on the Jaguar keypad, with '5' being dead center. This provides 9 initial views, but it doesn't end there. Rather than the camera moving with the ship, the camera remains in it's location on the plane, and tracks the ship by turning, providing a relistic and impressive view change in-game. (The actual target point is located some distance in front of the player's ship, so that the enemies are always visible). The distance from the plane to the rear of the ship is dependant on the ship's current x-y movement speed (based on

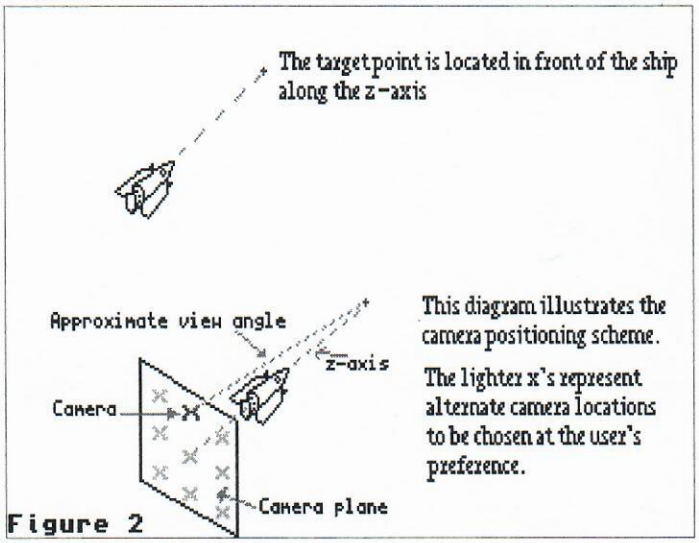

Figure 2

MAX(delta X, delta Y)). The effect is that the camera will smoothly track the ship while maintaining the perspective angle, and will zoom out as needed to track rapid movement. Camera view will also change during non-interactive scenes for effect (ie: swooping the camera around during boss introductions/explosions). This is simpler than it may sound, see fig 2.

The game itself will be polygon based, with texture-mapping and pre-rendered bitmaps used where it presents a better effect (possible 'for instance': explosions. Experimentation will be done to see what looks best.)

The player's ship, the Snowball, will come equipped with shields to protect from enemy attack. The shields will wear down and it is hoped to show the damage on the ship (in the form of marks and smoke, when it's really bad). Repair icons and weapon power-ups will also exist, along with several types of weapons.

The game is currently planned to be approximately 12 'stages', although official count is not actually given in the game - the player simply goes from one situation to the next. Depending on memory restrictions, it is hoped to have short animations explain the story as it goes along.

Finally, there are a good number of 'hidden diversions' planned. They are not essential to the game, and most will not help, either. They are intended as amusing 'bonus rounds' or intermissions in the action. The player must strive to find them by performing a specific action at a specific time, or can just sit back and enjoy an intense shooter.

A more Detailed Plot

The story opens as in the intro story, and this is all the player is given as background information. Things look pretty hopeless for Bob right from the start, as the player is faced with a lot of enemy drone ships, and only basic weaponry. After a fairly short time, the player will encounter the 'destroyer ship', the first boss, and be faced with an onslaught of weaponry to avoid.

The first boss is not impossibly difficult, to prevent too steep a learning curve, but some dodging will be required. A short time into the battle, however, the ship's computer will announce a localized subspace disturbance forming, believed to be a wormhole, and advise moving away a safe distance.

The enemy destroyer, picking up the same disturbance, will move away. The Snowball, however, will not be able to move away due to it's pre-programed flight/mission plan.

The wormhole will then open up onscreen, and pull the Snowball into it. A swirling mass of twisted light patterns, the wormhole will be a rather impressive sight. The player will be forced to dodge debris and electrical disturbance inside the wormhole as the Snowball travels forward through it.

When the end of the wormhole is reached, Bob finds himself in a temporarily unrecognized area of space. In communication with the mothership, however, he learns from his computer that he has been warped many parsecs to an area far behind the Qwertian front line. The mothership computer, having been programmed to send Bob as far into Qwertian space as he lives to reach, plots a course for the Qwertian's home solar system.

After some travel, during which Bob faces still more Qwertian battle drones, Bob reaches the solar system. Computer analysis shows that the five inner planets are generating a vastly powerful shield around the sun itself, preventing any radiation from entering or escaping the area where the sun should be, including sensor beams. Further readings show that the area around each shield generator is heavily monitored, and probably well defended. Bob, fitting into his role as Super Space Acer, plots to go in low from a distance to escape full detection, and destroy the shield generators. Before each scene, the computer is able to analyze and display the 'boss', including weak points.

The planets include a forested world, an ice world, a water world, a dead rock world, and a final world who's shield generator (and boss) reside in orbit around the planet. Each boss also gives Bob a new weapon system for the Snowball. Respectively; powerful but slow missiles, homing laser (weak but effective), a wave pulse weapon (which passes through enemies without stopping, whether it damages or fully destroys them), a powerful neutron pulse

weapon (fires a short distance and explodes in front of the Snowball), and an Anti-Gravity Repulse unit which pushes away and damages smaller enemies from the Snowball if they are close when it's fired (or pushes the Snowball away from larger enemies, still damaging them).

The planets may be done in any order, but the default order is as listed.

At this point the sun will open up from the shield, and sensors will show a large base actually within a cave on the sun, which turns out to be artificial. Bob flies straight into the fire, through a series of prominences and flares, as well as a fiery cave. A flaming guardian defends the end of the cave. Once it is destroyed, the cave, and the sun itself, begin to collapse (as is cheerily announced by the ship computer). At the same time, an icon for one last special fire weapon is released from the boss, but getting it and still escaping is a tricky thing to do.

The Snowball must now be maneuvered back out of the cave as quickly as possible, while fire pursues close behind. Once escape happens, the sun collapses, and explodes in a spectacular display of pyrokinetics.

As Bob begins to relax, the computer indicates the presence of an ion trail from an escaping craft, and a slowly closing 'jump point', sort of a controlled wormhole. Now determined to finish the Qwertians, Bob makes the jump.

Unlike the wormhole, the jump is more like a hyperspace leap, with streaking stars as well as colors. There is less debris, however, enemies 'jump' in with dangerous frequency.

At the end of the jump, the ship computer reports that contact with the mothership has been lost. A large destroyer vessel is detected, but no data on it is available, save for it's shape. Bob charges in again (hopefully with all his weapons ready!), and faces the final battle against a very powerful boss. (Naturally one would want the fire weapon here.)

Finally, all is done. The final boss is defeated. Bob manages to plot a course back towards Union space, and returns to see the destruction of the first destroyer that menaced him, by the Union fleet. The shockwave causes his own ship to stall, but he is rescued and towed back to base.

His story isn't believed, but checks out on sensor reports. The Admiral pretends it was always supposed to happen this way, and people cheer Bob, saying they knew he could do it. Secretly the admiral muses to himself that he's probably safer as a pilot anyway. Cue credits.

The Secrets!

Now, I've mentioned the secrets a couple of times, why not more often?

Simply put, the secrets are more humorous diversions that are added to the game for the fun of it. A lot will probably be thought of as the game develops, we come up with more all the time (some we discard, of course. Taste does have meaning here.)

The secrets are all accessed in different ways. They're all meant to be rather silly, and to draw a smile or two rather than aid the game.

For instance, some of the ideas we have under consideration follow:

-on 'Moving Target Software Designs' screen, a code will start the target man (from the Moving Target logo) running around the screen. The words will fade, and the player can use the 'sight' to try and hit the man. If the player gets him or a few seconds pass, the screen fades out into normal attract mode. (i.e.: no major screams or blood if the man IS hit.. just a quiet fade-out.)

-(poking fun at Jeff Minter, who we've all thought was a hell of a programmer for many years), another code in-game would bring up 'Llama Invaders', basically similar to Space Invaders (but with only 12 or 16 'aliens' and the Super Space Acer ship as the base), and replacing the aliens with Llamas similar to the ones in Mr. Minter's Llamatron.

-again, another code would activate a simple Asteroids-like game, but replacing the rocks with the faces of the developers (again using the SSA ship as the base).

-a side diversion was considered in which the SSA ship raced across the screen at high speed (with an accompanying faded yell), followed shortly by a pacman-like character making 'wokkawokka' noises

-an unusual keypress in certain space scenes could turn the stars into little chickens flying by

-if there is room for cinema-like displays, pressing down would lower a boom mike into the scene, which would be shoved by up with annoyance by a character

-the zeros in the score may, randomly, don a face and stick out their tongues

-a code on the first (undefeatable) boss may make him bulge his eyes and run in fear, complete with yelping sound

-the "classic Konami" 30-man code will work (but we won't call it that) [up, up, down, down, left, right, left, right, B, A, START]

-holding 'A' while pressing START (the other 'classic' code) will allow invincibility, level warps, and boss kills to be performed with 3-digit codes on the keypad

-randomly, on boot-up (or deliberately by holding 3 buttons), instead of Super Space Acer the title page will come up as 'The Adventures of Fluffy the Bear and Friends" (or something equally 'cute'), complete with 'cute' music. On button press, '(Ok, Just kidding)' comes up, followed by the proper intro.

Basically, nothing is meant to offend or infringe, it's all meant to be humorous. Anything that occurs randomly is also reproducible by button combinations or sequences. We get more ideas all the time, it's just a matter of how much space is left after the REAL game is in place.

Preliminary Numbers

This is a rather difficult section to be precise with, as a good deal of what follows is based on how much we can pull off with the hardware. I'll be optimistic, and try to be realistic based solely on what I have seen so far.

The player starts the game with 3 lives, and has a shield on each life. The number of hits the shield takes varies depending on the power of the weapon striking it. Shield repair icons are occasionally found, which restore a certain amount of shield energy.

There are approximately 12 stages. An estimated time of game completion for an expert player would be 30-45 minutes.

There will be at least 50 different types of enemy drone craft, hopefully more, and 8 large destroyer ships.

There are seven different types of weapons systems for the Snowball.

It's unreasonable to attempt to predict the number of polygons in the objects without actually working with the machine, but based on what I've seen there should be no problem making everything look good (i.e.: not chunky).

Again, frame rate is not predictable, but I am aiming for an absolute, worst-case scenario of 25 fps. Again, based on what I've seen, more should be reasonable to expect. 60 fps is the desired goal.

The simulated screen shots at the end, rendered with LightWave 3D, are provided to give a general idea of what selected screens may look like.

An undetermined number of secret diversions, based entirely on how much space is left in the cart.

Estimated Timeline

The following timeline for Super Space Acer - Jaguar is based on the average pickup time for new machines between the programmers, and allowing for general snags and delays. It is an **estimated worst-case** scenario.

Time	Expected Milestone
0 days	Receive Jaguar development system from Atari. Begin studying development manuals and, where needed, revising SSA design
10 days	Manuals reviewed, needed game revisions completed. Hardware has been played with whole time. (Can't learn a machine without hands-on!)
30 days	Have simple demos running flawlessly - bouncing balls, flashing colors, moving vectors, sound effects
45 days	Have got good feel for the hardware, main routines optimized, ready to code.
60 days	Minimum of main subroutines and functions in place, simple test routines running correctly
120 days	Main code completed or nearing completion. Debugging, touching up, and optimizing begins
160 days	Debugging nearing completion, final completion of level data and story data added. Secrets added where applicable.
200 days	Game all in place. Final debugging, optimizing and playtesting.

This plan is fairly reasonable, assuming of course that the Jaguar hardware doesn't expect the programmers to learn some wildly outrageous and unheard of concepts. For instance, if the math processor believes that

$$X+1 = \sqrt[3]{(X^5+3X)xC}$$

then there may be a problem. If not, all should be well. We have a good team of dedicated and intelligent programmers.

Who is Moving Target?

The following people currently make up our team. Each has a brief background.

Gordon Haddrell - *Supreme Dictator/Organizational Head* - Gordon is currently overseer of the team, and is in charge of organization. He likes to stick his nose into game design. He is also responsible for public relations, serves as a contact point in the team, and mixes a really dry martini. He worked for IBM for eight months and has a background in the retail video game retail market. His title is self-imposed.

Mike Brent - *Minion/Programmer* - the minion title assigned by Gordon, Mike is in charge of main programming and ensuring the software adequately comes together. Also sticks his nose unbearably into design. He has nearly ten years experience programming (mostly video game) in BASIC, assembly and C on a wide variety of home systems. Most projects, however, were not released. Some projects that did see release included a very early Super Space Acer for the Texas Instruments home computer, and the FlipSide BBS system, which was a complete operating system rewrite for the same machine (with supporting hardware). Has written countless small utilities for himself and friends, including a database program for the Kamloops, BC, High School, and a Pascal learning aid for the Summerland Secondary School. Has also written programs for the Apple 2, Atari ST, Atari 800, Vic 20, Amiga, IBM compatible and Timex Sinclair. Proficient in several dialects each of BASIC, C, and assembler (6502, 9900 & 68000), and also versed in FORTRAN, COBOL, and dBase. Recent projects have been aimed towards the Amiga, but work was greatly slowed by a four-year term in the armed forces. Enjoys coding to the point of obsession, and has single-handedly drank a six-pack of Jolt in one session.

[redacted] - *Musician* - a talented musician who has been composing for quite some time. Has worked on a wide variety of music projects, including the soundtrack for the computer animation 'War' produced by Digipen Computer Graphics Inc. He also hums a mean kazoo. A little bit normal, but he's good with music. He is available and familiar with composing specific themes to fit moods and events on demand.

[redacted] - *Programmer* - [redacted] has been a programmer for nearly 16 years on various systems, including the Commodore Pet, Vic 20, C64, Radio Shack Color Computers, Model IIIs, ICON networks, PC-6, IBM clones and the NeXT. Some of his projects have included a life simulation program (which won the IBM computing award of 1990), a program that displayed molecules from the chemical equation, a database for health and Welfare Canada, and an email handler for Ubitech Systems, Inc. Recently most of his programming has been recreational, and he has coded for several MUDs on the internet (Gateway, Dreamshadow and Torment.) Does likes to stick his nose into design, but not as often or annoyingly as Mike B. or Gord, and usually has useful suggestions. Predominant in several dialects of BASIC and several of C, also versed in FORTRAN and

Clipper. Has been known to fall down laughing on the spot at people in the street, especially very serious ones.

[redacted] - *Design/Quality Control* - [redacted] has been involved with video games for over 8 years, and has a firm grasp on what type of game is accepted and enjoyed. Nitpicky for the fine points, he's good at getting the programmers to get that little extra feature which does nothing, but looks good, in place. (The programmers secretly write animations at night depicting large objects like safes falling on his head as he walks down the street.) Also insistant on being big on design, he, Mike B. and Gord spend time arguing over what fits. Survives six hour sessions on the net, where he believes he's a dog with cosmic powers.

Steve Brent - *Design/Graphics/Suckup/Deranged Poet* - the latter titles, strangely enough, self-imposed, Steve's primary addition to the team is wacky ideas and design thoughts. He has nearly 6 years experience in game design, working side-by-side with Mike Brent. Less weird than he pretends to be, Steve has designed several simple games in the past for the Texas Instruments 99/4A (Monster Bash, Space Froggie) and the Atari ST (Battle Pong, Smalljaw the Unknown <unfinished>). He also designed the graphics for the original TI version of Super Space Acer. Still, he has been known to randomly name himself after objects found in his bedroom.

[redacted] - *Artist* - [redacted] is sixteen years old and has been drawing for at least 7 years, primarily super-hero characters. His art is of the highest quality, and will be a great asset to the team. A sample of his work is included, a character drawn on request recently for a friend. He is equally as adept at drawing on a computer with a mouse as he is on paper. Has recently trained a puppy to bark when told to 'speak'. (Also when told to 'sit', 'lay down', 'shake paw'...)

Moving Target Development Hardware

Moving Target currently has the following hardware available for development:

Amiga 2000 with 68030@25mhz, 7 megs RAM, 300meg HD
Amiga 2000 with 1 meg RAM, 130meg HD
Atari 1040ST with 1 meg RAM, 80 meg HD
Atari Mega STE with 4 megs RAM, 245 meg HD
Atari TT030 with 4 megs RAM, 130 meg HD
PC clone 486/66mhz with 8 megs RAM, 540 meg HD

Additional hardware includes: stereo audio samplers, scanner, color video digitizer

Software available: Lightwave 3D (Amiga), POV (ST and Amiga), several tracker programs (all platforms), SAS/C (Amiga), Turbo C (ST and IBM), Devpac Assembler (ST and Amiga), graphics manipulation software (all platforms), Digital Sound Studio (Amiga), SoundLab (ST)

So Is That All?

The answer is a resounding **NO!**

This document has detailed only Super Space Acer, but we have several other game ideas being developed. There is no intent to do one game, then sit back and relax. (We don't think Mike B. is capable of relaxing anyway.)

Currently under design are also a mech-warrior type of game, similar to Iron Soldier but with much greater flexibility and more detailed storyline, an as-yet unnamed Role-Playing Game (RPG), and a 3D battle-car simulation game. Depending on the speed of their design, one or the other will be ready to start as soon as Super Space Acer is finished.

After all that, all that's left is to beg, please, please, please, send a Jaguar DevKit! Look! Mike B is already chewing on his fingernails! He doesn't know when to quit! He'll hit his fingers soon! [redacted]'s on the ground laughing at him! Oh oh! [redacted] is barking at [redacted], who's picking up a rubber band and twanging it with his toes! Ack! Steve is reciting his poetry! Gord just stood up, and demanded that everyone stop and bow to him! Oh, the humanity! You can stop this! Please help!

Ok, that was silly. But you get the point. We are serious in our commitment to deliver quality games, we all think the Jaguar is a heck of a machine, but is in need of some killer software. That's where we come in. We have dedication, experience, motivation, and a sense of humour. We hope you'll agree and give us a chance to strut our stuff.

Overview of Video Tape

The following is more or less a segment-by-segment overview and explanation of the content of the video tape sent with this package.

This tape has been created and prepared in accordance with verbal agreement between Gordon Haddrell and Atari Corp, and all information contained herein and on the video tape is protected by the NDA between Atari Corp. and Moving Target Software Design.

All programs on this tape were written entirely by Mike Brent, all ideas, concepts, images, etc, are by Mike Brent, and do not necessarily reflect the ideals of Moving Target. Also remember that I did not have artists or musicians when I did these. <smile>

This document was written by Mike Brent. As mentioned, there is very little eye candy. However, there is also a floppy disk enclosed which contains a few source codes. Some programs mention 'Mike Ward', my old 'nom de plume', as well as 'Seahorse Software' and 'Julius Software', both names I used. Much of my older work is difficult to come by, as I've sold systems I once owned. I don't have as much recent work due to my 4+ years in the armed forces, but I've kept my skills up on smaller programs and 'quick-n-dirty' utilities.

Note that some of these programs do require external code, but I may not have included it. Plain data files, or selected external files, including header files, are not really required to demonstrate programming ability.

- - -

Texas Instruments TI-99/4A Home Computer - this was my first machine, and most of my software was written on it. I sold my TI in early '92, and thus I lost most of my software. I later bought another TI, however, I was only able to get back software which I had uploaded to bulletin boards, and even then, no source code. The source code I did manage to get was hard to come by. For the making of this tape, it took me over 35 mins to get the old TI working again... <grin>

A little background to help you gauge what you see against what the machine could do. The TI was a 16-bit computer powered by a TMS9900NL processor. The processor ran at approximately 4MHz. However, everything except the system ROM and 256 bytes of RAM was on an 8-bit data bus. The memory map contained an additional 32k of RAM, broken into two banks, one 24k, and one 8k. The 8k bank was used by c99 for stack space, and by the system for utilities and system data. Thus, only a portion of it could be safely used. There is also a separate 16k of video memory, accessed through the system as a memory-mapped device. The video processor was the TMS9918A, which supported 15 fixed colors (plus 'transparent') at 256x192 resolution. It supported 32 single color sprites with two sizes: 8x8 and 16x16. A magnify bit was also available which doubled the sprite size (but not it's resolution). Four sprites could be displayed on a single scanline. The primary graphics mode was character-oriented, 32 characters wide by 24 characters high. Another mode, which I used only for title screens due to speed, was a bitmapped mode which allowed the full 256x192 resolution, with the sole limit that every 8 pixels in a single row could contain only 2 colors. The audio processor was a 4-channel sound generator. The first three channels could generate only pure square-wave tones between 110hz - 44kHz. The fourth channel could generate only noise, with 3 preset rates of white noise, plus one definable through the third audio channel, and 3 preset rates of periodic noise, plus one definable through the third audio channel. Volume control per channel was four bits.

Whew!

- - -

Super Sled Acer - Super Sled Acer was written after Super Space Acer was conceived. It was intended as a joke, Gordon Haddrell and I had came up with a whole list of Super S* Acer games (Super Speed Acer, Super Stunt Acer, Super Street Acer, etc...). I did this one because it was the simplest. In this game, you drive Super Sled Acer down Doom Mountain on your new Super Sno-Ski Sled. We got the idea from looking at a GT Sno-Racer, and noticing a warning label on the brakes. I incorporated that into the program as a brake warning gauge, with the brakes failing if you held them too long. Every level includes a new opponent, being trees, rocks, skiers, snowmen (who move), snowmobiles, jumps, snowslides, tanks (who fire), and missiles. :) It was **supposed** to take only a few hours, but ended up taking me several days. This was due to an annoying bug in the sprite handler which I later found was actually an undocumented hardware 'feature' dealing with the moving of the 'hot-spot'. A couple of years later I cleaned the program up and re-released it, but I haven't seen source code for this program for over 3 years now. It was written almost entirely in c99, which was a version of Small-C for the TI, missing some of the more advanced features. (like LONG or FLOAT variables, or structures of any kind.)

Super Space Acer - this is the original game idea, and is **much** simpler in design than the current plan. (As reference to the current plan we have already submitted will show, the basic storyline is the only thing in common.) This version of SSA took me about a month in-between working shift-work for the armed forces. It was about 50/50 c99 and inline assembly. It was also my first attempt at writing music for a game, and is a large part of the reason we have a musician. <grin> The game was released as 'commercial', with a simple demo version released free. It was picked up by MediaWare in Florida for distribution and did reasonably well at the Chicago TI Faire in 1992. It received a couple of good reviews in several newsletters, and the monthly TI Magazine Micropendium also gave a fairly good review, commenting favorably on the graphics and the gameplay. The game has five levels, each with more and more difficult enemies, and a different and stronger boss. Source code is probably in the same place as source for Super Sled Acer.

Julius Demo - I wrote this demo after buying an Atari ST and finding I really liked graphic demos, and wanted to see what I could pull off on the TI. By re-writing the interrupt music driver to one that would handle simple ADSR, the music quality was greatly improved over Super Space Acer. I didn't write the tune, however, it was originally a Protracker music MODule called Kawai-K1, and I've no idea who wrote it. (I downloaded it.) The scrolltext doesn't mean much anymore, so I didn't run the whole thing on the video tape. The scrolling blue background was an attempt to see if I could simulate an effect I saw in a Genesis game, and the scrolling in the 'JULIUS SOFTWARE' letters was a twist on the same idea. The flashing border is just color-cycling. The flying balls at the bottom are sprites running through a pre-calculated sinewave table, and the audio bars are simply reading the current volume level from the music subroutine. This demo was written in one sitting, approximately 8 hours. It is over 90% assembly, with the C being more or less a shell to set up the screen and initialize the graphics. Source code is again nowhereville that I know of. Never sell off your own work, even if you sell the machine, I guess.

Super Space Acer II - I started this program shortly before I sold my TI. Although I didn't get past the title sequence, I pulled off a few things that even I thought were impossible. The animated intro screen is solid cheating, and isn't much of a feat, but the smooth scroll text is a feature that I have never seen on any other TI program, as the CPU is just too slow to scroll the bitmap screen like that. Most of the subroutines utilize a fair bit of assembly to pull off my tricks. This program **does** have source included, since I actually kept the disk. **(SSA2_C.TXT)**

FlipSide BBS System - this is a collection of files in **FS.ZIP** on the floppy disk, and is not on the video tape because I no longer have a system capable of running it. This software is written mostly in c99 v4.0 starting in late 1991 and continuing through until late 1992, when I sold my TI. I ran a BBS system called 'FlipSide' which was on-line continuously for that time period. I didn't produce much else because I spent most of my time upgrading and debugging the system. The version included turned out to not be my final

version, but I was happy to find this at all. The system was simple in nature, primarily message-based. There were also simple XMODEM file transfers (utilizing someone else's XMODEM subroutines), but since the system ran on two 180k floppy drives and one 196k hardware RAMdisk, files were not a priority. Most users liked the system, as it had personality, and was very reliable. Many people expressed surprise to me that it did not run on an IBM. I stopped using it so I could run commercial software on my Atari ST, so that I could use the hard drives for files.

- - -

Atari ST - I expect I wouldn't have to explain very much, and I only have one program to demo here. Although I wrote a few programs for it, I didn't keep the machine very long (due to finances). The only program I still have is presented here. I did do two or three programs in 68000 assembly, but since I have no trace of them it's unprovable. They were utilities, one was a Directory MOD player for the STF (there were lots for the STe, but not the STf, and I wanted one). The others were just practice opening windows and the like in GEM.

- - -

Flipette Chat - I've been toying with ideas for an AI-based chat program for a very long time, and every so often I take a crack at it. The TI was not really suitable for this, being slow and having little memory, so I tried it on the ST. The program was written in STOS Basic, mostly for the sake of easy and quick debugging. Although it fell short of my original design plan, I finished most of it off and released it. For a short while I ran it on my BBS while the BBS was on the ST (graphics did not transfer, of course). This had the effect of confusing people new to BBSing into thinking they were talking to the SysOp. <smile> The program was normally run compiled, but the old STOS compiler would not run on the Mega STe used to tape a sample run, so it's a little slower than usual. This program works by parsing the input line in several passes, checking for synonyms, keywords, and pronouns, modifying the string, and choosing an output string based on the keyword. If a keyword is not found, the program will check if a previous keyword has been 'talked out' yet. The winking eye is handled by one of STOS's simulated sprites, and the mouth is animated using two pages of graphics, and copying back and forth. Source is included as **FCHAT.BAS**

- - -

Miscellaneous - the next two programs are text-based, wholly C, and were compiled and run on several systems. The general intent of each was for use on a BBS system. I apologize for the poor video, composite video is hard-pressed to give clear 80-column text.

- - -

Tarot Reader - this program does a simple tarot card reading, by randomly choosing a card and an orientation (upright or reversed (upside-down)) from the database of the full 78 cards, and printing the meaning. Each card is also rated for being positive or negative, so that the computer can try to gauge an overall result. It was first done for the TI, and then for the Atari ST, Amiga, and IBM. The TI and ST versions were slightly different from the source presented here in **TAROT.C**. (Each was also named after the machine it was run on... Tee Eye, Ess Tee, Amoeba, and Eye Bee Emm.) It has run on each generation of my BBS (FlipSide), as well as running on another Amiga BBS in Ottawa known as Shockwave.

Creature of the Maze III - this is a gradual evolution of a program I first wrote in TI BASIC. COTM2 is included with the FlipSide source, but this program's source is **COTM3.C**. This program has run on at least two Amiga BBSs (plus my own), and has run on another two IBM boards. It is loosely D&D based, allowing the player to wander a large (75x75x10) maze, finding objects, meeting creatures both friend and foe, and having small puzzles to solve. The ultimate goal is to locate the 'creature', hidden somewhere on

the 10th level, where the enemies are most numerous and dangerous. If the player is able to defeat the creature, he gets to 'become' the boss, and his player's stats are saved as the new boss. Some of the objects were designed to be silly, like magenta pickles and batteries that don't fit anyway. There are also some silly spells ('A Spell Only a Fool Would Cast' will injure the caster), and silly enemies (Like '57 Chevies and Devil's Food Cakes). The game was meant to be as much a spoof as a serious game.

- - -

Amiga 2000 - the Amiga is a 680x0 based machine with a custom chipset to handle most aspects of system operation. The video processor can display up to 4096 colors at 320x200 (plus overscan) in a special Hold-And-Modify mode (useful mostly for still images), 64 colors at 320x200 (plus overscan) in a special 'extra-halfbright' mode which makes the second 32 colors a darker version of the first 32, 32 colors at 320x200 (plus overscan), or 16 colors at 640x200 (plus overscan). Interlacing is available to double the vertical resolution. The blitter, video chip, and audio processor can only access data in the first megabyte of RAM. The audio processor has 8-bit resolution and four voices, and can perform DMA to play sampled sounds at various rates through hardware. It has proven to be a difficult machine to find development software for, too. <grin> I've owned the Amiga for a few years, but only within the last two years have found enough information to begin programming graphics programs.

- - -

Fighting Game - No, really! That's the name! I was tired of all the silly names for fighting games that have been coming out, and since I wanted to do my own, I named mine Fighting Game. Originally planned a LOOONG time ago for the Lynx (yes, the Atari Lynx! I still have the manual I ordered for it. <grin>), my military contract prevented me from getting a devkit for that machine, and the design spent a long time being tossed around and re-planned for the TI (that idea didn't last long), the ST, and fell onto the Amiga. It was meant to be a slightly silly fighting game, no special features save that my friends were the characters (which I admit limits it's appeal a bit) and it's quality, as I've found a lot of fighting games have too much flash and not enough control. Nowadays another fighter is something that's less appealing than it once was, but I still like the idea. The character onscreen is actually Dan Cormier, playing 'Tatsuji', the 'Ryu-Clone' who is only in the fight for the money. Everything seen onscreen was written in one week, including the time required to learn to load an IFF image, use the hardware blitter, and call a sound utility. It was SUPPOSED to be a demo for our meeting at the E3, but I left the tape. <I banged my head on the dash of the truck for two days, and spent only the first day of the drive down in blissful ignorance.> The background is a 16-color picture, and the character himself is an 8-color character. Although he could have been any 8 colors, I left it greyscale as originally digitized by my cheap digitizer. An alternate palette is also available for the character, which is shades of red in this demo. The screen is a 32-color display, allowing each character it's own 8 colors. The program is also expandable, utilizing a programming language of my own creation to control the character (including the computer AI.) **FG.ZIP** contains the programs sources involved. Currently this program looks at another delay as I consider moving it to the IBM so I can give a lot more colors to the characters, especially since I have a better digitizer now. :)

Ami99 - I'd wanted a TI emulator so I could play around with my old TI without the hassle of hooking it up, and I finally started it. This is about a month of programming around shift work, however, many weeks of debugging went into it. I had a hard time finding a bug that apparently crashed the emulated TI, yet all the code looked right and the simple debugging output looked correct. Finally I added enhanced debugging output, and while things still looked okay, I noticed a few strange values being loaded into the VDP registers. Quick fix in the VDP program (which runs as a separate task, multitasking) to allow wrap-around within the 16k address bus (see the description of the TI, above), and it suddenly worked! I later learned that the video chip actually performs some internal masking, which allows a few little tricks to be pulled off in certain modes. Another undocumented (by TI) feature of the TMS9918A which has caused me to hunt through my code for bugs.... :) Anyway, this program has the CPU portion operating properly,

if slowly. The VDP needs to be re-done to support the various graphics modes and sprites, and the hardware access is not programmed yet (the CRU handles that, to anyone reading the source). The final intent is to replace most of the functions with assembly language for speed's sake. For demonstration, the emulator runs the TI ROMs and GROMs. In just the master title page appearing, the Amiga is running the TI 9900 code in it's ROM, which is then running the TI boot sequence in GROM. (GROMs contain a language called GPL, which is interpreted.) In order to demonstrate further, I start the loaded cartridge 'A-Amaze-Ing', which is also in GPL). No actual TI cartridges were actually raw ROMs, and I don't have any 3rd party cartridges, so I can't test to see what kind of speed that might give. :) See **AMI99.ZIP** for the sources to this program.

- - -

Finally, I threw on a silly little animation that I created. Not much programming, except to build some of the images, and again, I'm a programmer, not an artist! (cue Kirk... <grin>). Gord, the 'Supreme Dictator', says we can't use it because the logo is backwards. Next animation I build has **him** spinning inside a cube, faster and faster and fasterandfasterandfaster_{andfasterandfaster....}

Oops... went off track a bit. <grin>

Comments on Jaguar Project

I thought I would take the opportunity to answer a few questions which have been passed on to me through Gordon Haddrell. I'm not sure which ones have been resolved since, so I will just attempt to reassure that we are not completely out in left field. <grin>

It seemed to me that a good number of the questions asked were of a sort that Gord is not really in a position to answer, as they get down to the programmer, and this is not his job. I will be happy to take the phone for a Q&A session, if desired.

The major question seemed to center around our time-line, offering "only" 200 days from receiving the devkit to final optimization and play-testing (note: not necessarily finished product, just very close). I was reluctant to include a timeline, however, we were told to include one in our plan. We had to create the plan with nothing more than the public domain knowledge of the Jaguar, and little bits and pieces garnered from talking to other programmers at the E3. Taking this limited knowledge, and applying it against previous experience between team members in picking up new systems, we felt that the 200 days would be reasonable. Apparently the gentlemen who spoke to Gord felt this was outrageously short, apparently even beating out Id's conversion of Wolf3D. I'd like to offer a few comments in our defense:
- first, I'm dedicated and regaining my addiction to coding... I'm not talking 8 hour days... 16 hr sessions are not uncommon for me. Dan Cormier is well accustomed to picking up new systems and is a fast and efficient coder. We both produce code that requires little debugging.
- some of the figure was based on rumor as to what utilities come with the development kit, and assumption that little-to-no modification would be required to make them suit our needs.
- finally, a vast majority of Super Space Acer is 'eye-candy', the game itself is not as complex as it may sound. Most of the programming time will be adding special effects, fine-tuning game play, and tweaking the overall appearance.

The question was asked: Where are [we] going to get the polygon routines? Simple: if not included in the devkit, we will code them. Same with any other question about where we will get utilities.

'How many colors?' was asked. Well, we'd certainly love to do it in truecolor. Can we? We'll know after some playing around with the machine. 'So colors are the first thing to go?' was the follow-up. That, again, depends on the system. We ask back: can we squeeze truecolor by trimming such-and-such an effect here? Is the tradeoff worth it? We're hoping to have as few tradeoffs as possible, but the ultimate criteria is "what looks/plays better?"

Gord was asked if we realized the Jaguar's memory limitations. As my experience shows, I'm quite used to working in very tight memory environments, and I don't feel the Jag's RAM to be a limitation so much as a luxury!

Have our artists ever worked in true-color? Our main artist does most of his work on paper, so I'd say yes. However, he has proven himself equally skilled with a mouse. Whichever he feels more comfortable with is what we'll go with, we have scanners and digitizers. Again, of course, most of the graphics in SSA are rendered, not hand-drawn, so it's a minimum concern.

As I understand it, Gord was able to answer most of these questions, pretty much as I did, but I thought I'd elaborate as the head programmer. If there are further questions, I am more than happy to speak with whomever wants to ask them.

Mike Brent
Minion(?)/Head Programmer
MTSD