Stella

A multi-platform Atari 2600 VCS emulator

Release 1.0

Users Manual

1. INTRODUCTION

The Atari 2600 Video Computer System, introduced in 1977, was the most popular home video game system of the early 1980's. Now, thanks to Stella, you can enjoy your favorite Atari 2600 games once again!

Stella is a freely distributed multi-platform Atari 2600 VCS emulator; originally developed for Linux by Bradford W. Mott. The Stella emulation core is written in C++, which allows it to be ported to other operating systems and architectures. Since its original release versions of Stella for AcronOS, AmigaOS, DOS, FreeBSD, IRIX, MacOS, OpenStep, OS/2, Unix, and Windows have appeared.

Features

- Supports high speed emulation using optimized C++ code
- Includes several new games for the Atari 2600
- Supports high quality sound emulation using Ron Fries' TIA Sound Emulation library
- Emulates the Atari 2600 Joystick Controllers using your computer's keyboard or joysticks
- Emulates the Atari 2600 Keyboard Controllers using your computer's keyboard
- Emulates one Atari 2600 Paddle Controller using your computer's mouse
- Emulates the Atari 2600 Driving Controllers using your computer's keyboard or joysticks
- Emulates the CBS Booster-Grip Controller using your computer's keyboard or joysticks
- Supports cartridges using Atari's standard 2K and 4K format
- Supports cartridges using Atari's 8K, 16K, and 32K bank-switching schemes
- Supports cartridges using Activision's 8K bank-switching scheme (Robot Tank and Decathlon)
- Supports cartridges using CBS Electronics' 12K bank-switching scheme
- Supports cartridges using Chris Wilkson’s Megacart 128K bank-switching scheme
- Supports cartridges using M-Network's 16K bank-switching scheme
- Supports cartridges using Parker Brothers' 8K bank-switching scheme
- Supports cartridges using Tigervision's bank-switching scheme with up to 512K ROM
- Supports Supercharger single-load and multi-load games
- Supports a property file for setting the properties associated with games
- Built-in property file contains properties for over 540 games
- Supports slowing down or speeding up the emulator to a specified number of frames per second
- Supports the NTSC and PAL television palettes
- Supports several "undocumented features" of the TIA graphics chip used by some games

- TIA emulation supports full collision checking
- DOS version of Stella works with most emulator game shells such as "Game Menu" by Jim Pragit
- DOS version of Stella supports the standard 320x200 VGA graphics mode as well as the extended 320x240 graphics mode

2. WHAT YOU NEED

The following is a list of basic requirements for running Stella:

DOS

The DOS version of Stella is designed to work on an IBM compatible PC with the following requirements:

- DOS 5.0 or higher
- 8 MB of RAM
- 256 color VGA graphics card
- 486/120 MHz required; 586/166 MHz highly recommended
- Sound Blaster or 100% compatible sound board required for sound
- Joysticks or gamepads are highly recommended
- Mouse and mouse driver required for paddle emulation

If you'd like to do development work on Stella you'll need DJGPP with the GNU C++ compiler version 2.8.1 as well as the *make* utility. You may find it useful to have *sed* and the *m4* macro processor as well.

Linux

The Linux version of Stella is designed to work on a Linux Workstation with the following requirements:

- Linux Kernel 2.0.x
- X Window System
- GNU C++ compiler version 2.7.2 and the make utility are required for compiling the Stella source code; GNU C++ compiler version 2.8.1 is highly recommended
- 16 MB of RAM required; 32 MB highly recommended
- 8 bit color graphics card required; 16 bit color graphics card highly recommended
- 486/120 MHz required; 586/166 MHz highly recommended
- Open Sound System with a supported sound card required for sound
- Joystick device driver required for joystick support

If you'd like to do development work on Stella you may find it useful to have *sed* and the *m4* macro processor.

Unix

The Unix version of Stella is designed to work on a workstation with the following requirements:

- Unix operating system
- X Window System
- GNU C++ compiler version 2.7.2 and the make utility are required; GNU C++ compiler version 2.8.1 is highly recommended
- 32 MB of RAM
- 8 bit color display required; true color display highly recommended
- High performance CPU
- Open Sound System required for sound

If you'd like to do development work on Stella you may find it useful to have *sed* and the *m4* macro processor.

3. INSTALLATION

Once you've obtained a distribution you'll need to install it on your system. You'll find instructions for installing Stella under various operating systems below. If your operating system isn't listed then see the installation instructions included with the distribution for your system.

DOS

The DOS version of Stella is distributed as a ZIP archive containing the DOS executable as well as other files. You can install it as follows:

1. Change directories to the root directory: `CD C:\`
2. Extract the Stella distribution:
   `unzip st`*release*`.zip`
3. Add the following line to your `AUTOEXEC.BAT` file:
   `SET PATH=%PATH%;C:\STELLA`
4. If you have a Sound Blaster-compatible sound card make sure the `BLASTER` environment variable is set in your `AUTOEXEC.BAT` file. For example:
   `SET BLASTER=A220 I7 D1`
5. If you have a mouse make sure your mouse driver is loaded in either the `CONFIG.SYS` file or the `AUTOEXEC.BAT` file
6. Reboot your system

For the second step you'll need a program for extracting ZIP archive files. You can use PKUNZIP by PKWARE, Inc. for MS-DOS, the Info-Zip UnZip tool for MS-DOS, or a number of other programs available for extracting ZIP files.

Linux

The Linux version of Stella is distributed as a compressed tar file containing the Linux executables as well as other files. The executables were built on a machine running Red Hat Linux version 5.0. If the executables do not work on your system then download the Unix source code distribution and build them yourself. To install the distribution do the following:

1. Extract the files from the distribution:
   `tar -zxvf stella-`*release*`-linux.tar.gz`
2. Login as root and change directories to the distribution directory
3. Copy the executables to a system directory:
   `cp xstella stella-sound /usr/local/bin`

If you have a joystick module installed, such as `joystick-0.8.0.tar.gz`, you should be able to play games using joysticks.

Unix

The Unix version of Stella is distributed as a compressed tar file containing the C++ source code as well as other files. The source code can be compiled under most Unix operating systems as well as DOS. The Unix code was developed with the GNU C++ compiler version 2.8.1, however, it should compile with other C++ compilers. The DOS port was developed with DJGPP using GNU C++ version 2.8.1.

1. Extract files from the distribution:
   `tar -zxvf stella-`*release*`-src.tar.gz`
2. Change directories to the `stella-`*release*`/src/build` directory
3. Edit the `makefile` to meet your needs
4. Type `make` and follow the on screen instructions to build the `xstella` executable
5. Install `xstella` in a directory that's in your path (`/usr/local/bin`)
6. Change directory to the `stella-`*release*`/src/ui/sound` directory
7. Type `make` and follow the on screen instructions to build the `stella-sound` executable
8. Install `stella-sound` in a directory that's in your path (`/usr/local/bin`)

Currently, sound is supported on systems using the Open Sound System. Sound has been tested and is known to work under Linux and BSDI. For additional information on OSS see the 4Front Technologies web page at:

```
http://www.4front-tech.com/
```

If you have some free time please try to port `stella-sound` to your favorite version of Unix.

4. GAMES

In order to play an Atari 2600 game with Stella you need a ROM image of the game. A ROM image is a file that contains the data from the game cartridge or cassette.

Cartridges

Most games for the Atari 2600 were stored on cartridges. These cartridges usually consisted of a single ROM chip, which had the computer program for the video game stored on it. Plugging the cartridge into the Atari 2600 gave the 2600's microprocessor access the program stored on the cartridge.

In a similar way you have to "plug" a copy of a cartridge into Stella when you want to play it. Having a ROM image, `BIN` file, of the cartridge allows you to do this. A ROM image is a file, which contains the actual data read from a cartridge. There are several ways to obtain a ROM image of a cartridge:

1. If you're handy with a soldering iron then you can design and build a device that plugs into the printer port of a PC and read the data from the cartridge
2. You can purchase one of the Atari 2600 Action Packs by Activision and use the ROM images from it
3. You can search around the internet and find some ROM images to download

WARNING: It is illegal to use ROM images of games that you do not actually own since these games are still copyrighted.

Supercharger Cassettes

Supercharger games were not stored on cartridges instead they were stored on cassette tapes. The Supercharger, which plugged into the Atari 2600's cartridge slot, loaded games into it's 6K of RAM using a standard audio cassette player. The Supercharger also supported multi-loading, which allowed games to be broken into several 6K pieces and loaded at different times. This was useful for large games, which had distinct parts such as role playing games.

Most Supercharger ROM images are stored in an 8448 byte file. However, ROM images of multi-load games are stored in a set of 8448 byte files. The names of these files have a two character sequence number in them which indicates what load they are. The sequence starts with zero, skips a few numbers and then increments by one.

Stella supports multi-load games, however, the set of ROM images must be combined into a single ROM image file. For example to create a multi-load ROM image file for Survival Island you would do the following under Unix:

```
% cat survivl0.bin survivl6.bin survivl7.bin > survivl.bin
```

Once you have the multi-load ROM image file, `survivl.bin` in this case, you can play the game using it.

5. PLAYING GAMES

Once Stella is installed and you have some ROMs you're ready to start playing. To play a game following the directions for your operating system given below.

DOS

The DOS version of Stella uses command line arguments to specify the game you'd like to play as well as other options. To run Stella use a command line of the following format:

```
stella [-fps n] [-modex] [-paddle n] filename.bin
```

Options

-fps n	Attempt to display *n* frames per second instead of the default 60 frames per second (if the computer isn't fast enough then it may be unable to display the desired number of frames per second)
-modex	Indicates that the 320x240 graphics mode should be used instead of the default 320x200 graphics mode
-paddle n	Sets the paddle emulated by the mouse to paddle 0, 1, 2 or 3 (defaults to paddle 0)

Emulator Game Shells

Instead of using the command line to run Stella you may find it easier to use an emulator game shell. An emulator game shell allows you to pick games from a menu without having to type everything at the command line. One such game shell is Jim Pragit's "Game Menu" which can be found at the following URL:

```
http://members.aol.com/jpsoftco/gamemenu.htm
```

There are other game shells available, however, you'll have to search for them.

Keyboard Layout

The console switches and controllers for the 2600 are mapped to the computer's keyboard as follows:

F1	Select Game	*F5*	Left Player Difficulty B
F2	Game Reset	*F6*	Left Player Difficulty A
F3	Color TV	*F7*	Right Player Difficulty B
F4	Black & White TV	*F8*	Right Player Difficulty A
Esc	Exit Game		

Left Joystick

Up	*W* or *Up Arrow*
Down	*S* or *Down Arrow*
Left	*A* or *Left Arrow*
Right	*D* or *Right Arrow*
Fire	*Tab* or *Space*

Right Joystick

Up	*O*
Down	*L*
Left	*K*
Right	*Semicolon*
Fire	*J*

Left Booster-Grip

Up	*W* or *Up Arrow*
Down	*S* or *Down Arrow*
Left	*A* or *Left Arrow*
Right	*D* or *Right Arrow*
Fire	*Tab* or *Space*
Trigger	*1* or *Z*
Booster	*2* or *X*

Right Booster-Grip

Up	*O*
Down	*L*
Left	*K*
Right	*Semicolon*
Fire	*J*
Trigger	*N*
Booster	*M*

Left Driving Controller

Left	*A* or *Left Arrow*
Right	*D* or *Right Arrow*
Fire	*Tab* or *Space*

Right Driving Controller

Left	*K*
Right	*Semicolon*
Fire	*J*

Left Keypad

1	2	3	*1*	*2*	*3*
4	5	6	*Q*	*W*	*E*
7	8	9	*A*	*S*	*D*
*	0	#	*Z*	*X*	*C*

Right Keypad

1	2	3	*8*	*9*	*0*
4	5	6	*I*	*O*	*P*
7	8	9	*K*	*L*	*;*
*	0	#	*,*	*.*	*/*

Linux and Unix

The Unix version of Stella uses command line arguments to specify the game you'd like to play as well as other options. To run Stella use a command line of the following format:

```
xstella [-fps n] [-paddle n] filename.bin
```

Options

-fps n	Attempt to display *n* frames per second instead of the default 60 frames per second (if the computer isn't fast enough then it may be unable to display the desired number of frames per second)
-paddle n	Sets the paddle emulated by the mouse to paddle 0, 1, 2 or 3 (defaults to paddle 0)

Keyboard Layout

The console switches and controllers for the 2600 are mapped to the computer's keyboard as follows:

F1	Select Game	*F5*	Left Player Difficulty B
F2	Game Reset	*F6*	Left Player Difficulty A
F3	Color TV	*F7*	Right Player Difficulty B
F4	Black & White TV	*F8*	Right Player Difficulty A
Esc	Exit Game	=	Window Size

Left Joystick

Up	*W* or *Up Arrow*
Down	*S* or *Down Arrow*
Left	*A* or *Left Arrow*
Right	*D* or *Right Arrow*
Fire	*Tab* or *Space*

Left Booster-Grip

Up	*W* or *Up Arrow*
Down	*S* or *Down Arrow*
Left	*A* or *Left Arrow*
Right	*D* or *Right Arrow*
Fire	*Tab* or *Space*
Trigger	*1* or *Z*
Booster	*2* or *X*

Left Driving Controller

Left	*A* or *Left Arrow*
Right	*D* or *Right Arrow*
Fire	*Tab* or *Space*

Left Keypad

1	2	3	*1*	*2*	*3*
4	5	6	*Q*	*W*	*E*
7	8	9	*A*	*S*	*D*
*	0	#	*Z*	*X*	*C*

Right Joystick

Up	*O*
Down	*L*
Left	*K*
Right	*Semicolon*
Fire	*J*

Right Booster-Grip

Up	*O*
Down	*L*
Left	*K*
Right	*Semicolon*
Fire	*J*
Trigger	*N*
Booster	*M*

Right Driving Controller

Left	*K*
Right	*Semicolon*
Fire	*J*

Right Keypad

1	2	3	*8*	*9*	*0*
4	5	6	*I*	*O*	*P*
7	8	9	*K*	*L*	*;*
*	0	#	*,*	*.*	*/*

6. Game Properties

Stella uses game properties to specify the "best" settings for a game. Stella's built-in property file contains specific settings for over 540 games. If for some reason you need to specify your own properties for a game you can create a property file. The best way to do this is to download the latest `stella.pro` file from the Stella web site and add your new properties to it.

The correct location to place your properties depends on which version of Stella you're using:

DOS

This version of Stella looks for the property file `stella.pro` in the current working directory. If this file is found game properties are read from it; otherwise the built-in property file is used.

Linux and Unix

This version of Stella looks for the property file `.stella.pro` in your home directory. If this file is found game properties are read from it; otherwise the built-in property file is used.

Property File

A property file consists of some number of blocks. Each block in the file contains the properties for a single game. For example the general format of a property file is:

```
; Comments
"Cartridge.Name"  "Value"
"Property"        "Value"
""

; Comments
"Cartridge.Name"  "Value"
"Property"        "Value"
""""

…

; Comments
"Cartridge.Name"  "Value"
"Property"        "Value"
""""
```

Every block in the property file must have a unique value for the Cartridge.Name property.

Properties

Each block in a property file consists of a set of properties. Stella supports the properties described below.

Cartridge.Filename

This property indicates the name of the file containing the ROM image. In general this property isn't needed, however, if present Stella uses it while attempting to match a game with its properties (i.e. if the value of the property matches the filename of the ROM then it uses those properties).

Cartridge.MD5

This property's value should be the hexadecimal representation of the MD5 checksum for the game. Stella uses this property while attempting to match a game with its properties. If the value of the property matches the MD5 checksum of the ROM image then its uses those properties.

Cartridge.Manufacturer

This property should be the name of the game's manufacturer.

Cartridge.ModelNo

This property should be the manufacturer's model number for the game.

Cartridge.Name

This property should be the actual name of the game. It's very important that this property's value be unique from all of the other cartridge names in the property file.

Cartridge.Note

This property contains any special notes about playing the game.

Cartridge.Type

This property should indicate the bank-switching type for the game. The value of this property must be: Auto-detect, 2K, 3F, 4K, AR, E0, E7, F4SC, F6, F6SC, F8, F8SC, FASC, or FE. For more information about bank-switching see Kevin Horton's 2600 bank-switching document.

Console.LeftDifficulty

This property should indicate the default difficulty setting for the left player. The value of this property must be A or B.

Console.RightDifficulty

This property should indicate the default difficulty setting for the right player. . The value of this property must be A or B.

Console.TelevisionType

This property should indicate the default television setting for the game. The value of this property must be Color or BlankAndWhite.

Controller.Left

This property indicates what type of controller to use for the left player. The value of this property must be Booster-Grip, Driving, Keyboard, Paddles, or Joystick.

Controller.Right

This property indicates what type of controller to use for the right player. . The value of this property must be Booster-Grip, Driving, Keyboard, Paddles, or Joystick.

Display.Format

This property indicates the television format the game was designed for. The value of this property must be NTSC or PAL.

Display.Xstart

This property indicates the horizontal location to start displaying pixels at on a scan-line. The value of this property must be n such that $0 \leq n \leq 80$ and n is divisible by 4.

Display.Width

This property indicates the number of pixels to display per scan-line. The value of this property must be n such that $80 \leq n \leq 160$ and n is divisible by 4.

Display.Ystart

This property indicates the scan-line to start displaying at. The value of this property must be n such that $0 \leq n \leq 64$.

Display.Height

This property indicates the number of scan-lines to display. The value of this property must be n such that $100 \leq n \leq 256$.

Emulation.CPU

This property indicates the CPU emulation quality. The value of this property must be High or Low.

Emulation.HmoveBlanks

This property indicates whether the TIA HMOVE blank bug should be emulated or not. The value of this property must be Yes or No.

7. ACKNOWLEDGEMENTS

I started developing Stella during the fall of 1995 and since then I've received help from a number of people around the world. Some people have provided technical help while others have offered suggestions and praise. I am forever grateful for all the help and support I have received over the last few years. The following is an incomplete list of the people who have played a part in bringing Stella to you:

Piero Cavina	Allowed "Oystron" to be included in the Stella distribution
Bob Colbert	Allowed "Okie Dokie" to be included in the Stella distribution
Joe D'Andrea	Author of the IRIX port of Stella
Ron Fries	Author of the incredible TIA Sound library
Aaron Giles	Author of the Power Macintosh version of Stella
Mark Hahn	Allowed "Elk Attack" to be included in the Stella distribution and provided a description of TIA HMOVE blank bug
Kevin Horton	Author of the awesome Atari 2600 bank-switching document
Erik Kovach	Author of the property file for versions 0.7 and 1.0 of Stella
Daniel Marks	Provided code to improve the keyboard joystick support for Stella version 0.1
David McEwen	Author of the Acorn version of Stella
Jeff Miller	Author of the Windows version of Stella
Dan Mowczan	Provided a Supercharger to help with Stella's development
Jack Nutting	Author of the OpenStep version of Stella
Jim Pragit	Author of the "Game Menu" emulator game shell
Chris Salomon	Provided information and code to implement Supercharger support
Jason Scott	Organizer of the property file archive for early versions of Stella; now he's helping with the web site
Raul Silva	Helped with the design and graphics for an early version of the Stella web site
Chris Snell	Maintained a mirror of the Stella FTP site
Darrell Spice Jr.	Author of the OS/2 version of Stella
Eckhard Stolberg	Provided a description of the TIA bug that produces the star field effect in Cosmic Ark and provided the PAL television palette
Matthew Stroup	Author of the Amiga version of Stella
Joel Sutton	Author of the FreeBSD version of Stella
Greg Troutman	Allowed "This Planet Sucks" to be included in the Stella distribution
Keith Wilkins	Maintained the DOS version of Stella until release 0.7
Jeff Wisnia	Provided a technical data sheet for the 6532 RIOT chip
Countless others	Many of the features present in Stella are a result of input from people who've sent me suggestions

8. LICENSE & DISCLAIMER