Splotch Previewer

Tim Dykes - Ian Cant
University of Portsmouth

Contents

Contents	Page
Setup and Build process	3
Command system	3
Further Comments	5
Typical Usage Scenario	5
Adding new Renderers	6
Test Computers	7

Previewer Readme

1. Setup and build process

Before building please ensure you have the OpenGL development libraries and the X11 development libraries installed, plus necessary compilers. Particular packages will vary depending on OS, on Ubuntu you can use xorg-dev and libgl1-mesa-dev, plus build-essential for compilation.

The previewer can be activated by uncommenting the appropriate option at the top of the makefile, named the rendering method can then be chosen further down (>line 258). In the current build the previewer is already enabled, with the highest quality renderer chosen.

There are 3 different renderers to choose from, depending on your hardware capabilities, further instructions as to renderer choice and capabilities are in the makefile as well as described in the renderer section of this document.

To run Splotch with the previewer, run Splotch as usual but add the switch -pv to the command line.

2. Command System

The command input for the application can be activated using the hash key (#). This command console can be used to enter commands in to the system. The console input can be closed again with the hash key.

To move around the scene use the WASDQE key's to move forward, left, back, right, up and down respectively. To rotate (or orbit) around the centre point of the data using the IJKL keys, the centre point in this case is the centre of a computational box composed of the maximum and minimum particle locations on each axis. Using the mouse left click button the camera can be freely rotated, this allows a 360 degree field of rotation.

A number of commands can be input using the command line interface, bring up the console and enter the command using the keyboard. Once a command has been entered, it can be executed using the enter or return keys. Command input is not case-sensitive. A list of commands and their resulting actions is shown below:

Command input	**Resulting action**
quit	Terminates Application
write param file [filename]	Write current parameters to [filename] As standard splotch parameter file.par
set palette [filename] [particletype]	Set a colour palette for a particular particle type, as defined by Splotch particle types
reload colours	Reloads the colours, this necessary after setting new palettes and will reload the renderer. This will also reset the camera.
get fps	Output current frames per second, which is an average calculated every 0.5 seconds
set animation point [time]	Set an animation point at [time] seconds

remove animation point	Remove the previous animation point
set move speed [speed]	Set the camera movement speed (default 200) Movement and rotation speed are discussed further in comments below
set rotation speed [speed]	Set the Rotate around target speed (default 200)
preview animation	Interpolate and preview the currently loaded animation (there may be a delay while points are interpolated)
save animation [filename]	Save animation to [filename] in an internal previewer format, the layout of which will be described below
load animation [filename]	Load animation [filename] from internal previewer format
set xres [res]	Set the x resolution of the image (default value taken from parameter file, or 800 if unavailable in parameter file)
set yres [res]	Set the y resolution of the image(default value taken from parameter file, or 800 if unavailable in parameter file)
set resolution [xres] [yres]	Set both resolutions simultaneously
set fov [fov]	Set field of view in degrees
interpolate	Interpolate the currently loaded list of animation points, this will be done automatically when previewing the animation/writing an animation file
write splotch animation file [filename]	Write a Splotch geometry file (now called scene_file)
set camera interpolation [interpolation]	Set the interpolation type of the camera movement (linear or cubic)
set lookat interpolation [interpolation]	Set the interpolation type of the lookat movement (linear or cubic)
run [splotch executable] [parameter file]	Spawn a new process to run splotch with specified parameter file
view image [filename]	View a splotch rendered tga image
stop viewing	Return from image viewing to preview
set param [name] [value]	Set a parameter in the parameter file
get param [name]	Display a parameter from the file

3. Further Comments

To start the previewer using camera parameters specified in the parameter file, instead of recalculating position to view entire set, add recalc_preview_cam=false to the parameter file.

Particle radii are compensated within the previewer to provide a smoother blend, the default modifier is 2 and will be multiplied with the actual radius of a particle. To change this behaviour, add preview_radial_mod=1.0 to parameter file, where 1.0 is your desired multiplier.

The Read/Write Parameter file methods, as well as the Set and Get Parameter methods will be unaffected by changes made to the structure of the parameter file, for example any new parameter added to a .par file that has not been used before will still be loaded and written out, as long as the file remains as a string-string map.

Movement and Rotation speeds are arbitrary unit scales compensated for frame rate. As a rough guide for both scales - 50 is slow, 200 average, and 500 is fast.

When creating animations with the "rotate around centre" feature, it is suggested to only use linear interpolation for both camera and lookat. Default starting values for interpolation are linear for lookat, cubic for camera movement. Best results are usually achieved using the default interpolation setup unless rotating around centre.

Splotch geometry files created are saved to previewer/data/splotchAnimationPaths
Full filepaths and extensions when naming the file in the previewer on-screen command line are not necessary, as it will automatically look in this folder.

The run command expects the splotch executable name due to the executable potentially being named differently depending on makefile settings when it was built (eg Splotch5-generic, Splotch5-plx etc)

The set param command will not update the previewer with the new parameters, it will only update the file, so use specific commands for updating parameters used within the previewer - such as fov and resolution. This generic setter is for parameters unrelated to the previewer. Values updated using this method will still be written to file when “write param file” is called, just not updated within the current preview.

The view image command is relative to the executable directory, so files to view should be in there. This is where splotch saves them so there should not be an issue, but if you have saved it somewhere else remember to write the correct file path. You should be able to use it to view any splotch created tga regardless of current preview.

When previewing an animation, it will run at a different speed than expected. This is because the animation writer will create 30 frames per second, with the assumption that when Splotch is run with a geometry file created from this animation the final results will be composed into a film running at 30fps. The previewer itself is not limited to 30fps, and may at times be running faster/slower than this.

The internal previewer animation file is splitch into paths for each parameter being changed. Each

path begins with a path number, component type and parameter, then xyz coordinates seperated by hashes per animation point. Each path is ended with an @ symbol and a new line. Eg:

0,camera,camera_x,#0,-2574.2319,1,#5,-2574.2319,1,#10,-7241.7485,1,#15,-7241.7485,1,#@
1,camera,camera_y,#0,55747.3828,1,#5,48746.6875,1,#10,48746.6875,1,#15,45079.9219,1,#@

Commas delimit individual data items, hashes delimit triplets of data, @ symbols delimit paths. It should not be necessary to interact with these files.

4. Typical usage scenario

This is a description of how the previewer would typically be used.

Start out by ensuring the previewer option at the top of the makefile is not commented. Then scroll down to the previewer options and ensure the most suitable render has been picked, the highest quality one that works on the hardware.

Then build, and run the executable. In a typical case it is with the command:

./Splotch5-generic parameterfile.par -pv

This should open up the previewer window, with a full view of the data on screen. Activate the on screen command line interface by pressing the hash key(#). Check fps with "get fps" to see if it is running okay. Change any relevant parameters if necessary, for example the resolution could be changed with "set resolution 1920 1080".

Keyboard and mouse would be used to navigate the scene and find a nice starting point for the creation of an animation. At this point type the command "set animation point 0". This will set the starting animation point. Then navigate to another point in the simulation, and type "set animation point 5". This will create a second animation point at 5 seconds. Continue doing this until you have a suitable length animation, and then type preview animation. There will be a short delay while points are interpolated, and then the animation will play.

If you are not happy with some of it, you can remove points then add more ("remove animation point"). Once happy, you can call write Splotch animation file [filename]. This will create a geometry file for use within Splotch. To then run Splotch with this geometry file use "set param scene_file [file]" remember to put the whole filepath if you have not moved it to the same directory as the executable, so in the default case [file] would be:
previewer/data/splotchAnimationPaths/file.txt

To see a splotch image of what you are looking at, you can call "write param file filename.par" , then call "run Splotch5-generic filename.par" substituting the executable name with a relevant one if necessary. You can then call "view image imagename.tga" to see the image created, and "stop viewing" to return to the preview. The image name is usually already in the parameter file, but could be changed with "set param outfile filename" (no extension).

5. Adding New Renderers

To add a new renderer, one must implement the IRenderer interface (previewer/libs/renderers/). This

is fairly simple, and only involves inheriting from the interface and implementing 6 key functions, which can be seen on the class diagram included with this document. The Load function is where the renderer is provided with the particle data, Update, Draw and Unload functions are also necessary. You are also provided with event functions to react to key press/mouse motion events.

Having implemented the renderer, you will need to create a section in the makefile to add your renderer as a choice, build your particular renderer object and create the include. This is explained further in the relevant section of the makefile.

6. Test Computers

This a list of the hardware/software combinations this software has been tested and confirmed working on. If you find an issue with your setup, where no renderer will draw, or a segfault etc, please let us know. Note on computers with low memory allowances, the FF_VBO renderer is fairly memory intensive compared to the others, and larger filesets may cause problems.

Operating System	CPU	GPU	Drivers	RAM
Arch Linux 3.6.2-1	Intel i5	Ati Radeon HD5770	ATI Proprietary for linux (version unknown)	8G
Arch Linux 3.6.2-1	Intel i5	Nvidia GTX 670	Nvidia Proprietary for linux 304.51	8G
Arch Linux 3.6.2-1	Intel i7	Nvidia GTX 650M	Nvidia Proprietary for linux 304.6	3G
Ubuntu 12.04	Unknown	Nvidia Quadro 5000	Nvidia Proprietary for Linux 304.51	16G
Ubuntu 12.04	AMD Athlon II	ATI Radeon HD5670	ATI Proprietary for linux (version unknown)	4G