StimServer

Michael Stephan

May 31, 2016

Contents

1   Error Handling

Error state queries (introduced in version 1.1.5.0) are intended to detect anomalies in the server operation (or bad client requests). *This functionality depends on the detection of all kind of server state problems and badly formated client commands. This is not yet implemented and has to be integrated into future versions of the server.*

2   Stimuli

. . .

2.1   Particle Stimuli

Particle stimuli are composed of a given set of dots (particles) which move in a certain direction with a given speed. The positions of the dots (and optionally the direction of movement per particle) are read from a binary file. The first two rows (in Matlab) of the file contain the normalized x- and y-coordinates (single floats in the range $-1 \ldots +1$). The file may also contain a third row specifying the direction of movement for each particle (angles in degrees as single floats). Dots moving out of the rectangular viewport (specified at the creation of the object) wrap to the opposite edge. The viewport can be masked with a circular and/or Gaussian patch (see below).

To create a coordinate file for a particle stimulus object of 100 random dots with Matlab, you might use
```
NumericBinaryFile('temp.part',2.0*rand(2, 100, 'single')-1.0);
```
The particles of such a file all move according to the direction and speed parameters set for the stimulus object.

To create a similar file where each particle moves into a distinct direction (between -5° and 5° in this example) you may use
```
NumericBinaryFile('temp3.part', vertcat( ...
    2.0*rand(2, 100, 'single')-1.0, 10.0*rand(1, 100, 'single')-5.0));
```
In this mode the value of the direction parameter will be added to the angles read from the file, i.e. in the given example, if you set the direction parameter to 45, the particles will move in directions between 40° and 50°.

2.1.1   Patches for Particle Stimuli

You can use a circular and/or Gaussian patch. With a circular patch any particles outside a circle with a given radius are unvisible.

With a Gaussian patch the transparency of a particle will increase with it's distance from the center of the stimulus. The opacity value α of a particle with distance d from the center of the stimulus is

$$\alpha = e^{-\frac{d^2}{2R^2}}$$

where R is the "radius" parameter set for the patch.
. . .

2.2 Bitmap Brush Objects

"Bitmap Brush Objects" (introduced in version 1.1.2.0) provide a way to apply "Opacity Masks" to images. There are different modes depending on the size of the components involved.

If the width and height of the opacity mask are smaller than the respective dimensions of the bitmap brush, a centered rectangular part of the bitmap brush with the size of the opacity mask is used for the operation. You may shift the rectangle relative to the bitmap brush. This is the preferred mode of operation.

If the width and height of the opacity mask are greater than the respective dimensions of the bitmap brush used, a centered subset of the same size as the target is used as the mask. Making the opacity mask sufficiently large (twice the size of the bitmap brush) allows the mask to be shifted relative to the target.

You can replace an opacity mask by simply loading a new one. Bitmap brushes without an attached opacity mask render like normal pictures (but more expensive).

2.3 Pixel Shaded Pictures

Introduced in version 1.1.4.0 these objects extend the power of pixel shaders to pictures. These stimuli are created in two steps. The first step is to load a normal picture object. In the second step a pixel shader file is loaded. This second command has a reference to the key of the picture object on which the shader will operate. An internal copy of the original picture will be created. Therefore the picture object might be deleted afterwards (if it's not going to be used otherwise).

3 Animations

Animations change a certain property (e.g. position) of a stimulus object with each frame. Currently only absolute stimulus positions can be specified. The position set for the stimulus object is overwritten.

Animations start as soon as they are assigned to an enabled stimulus object. If an animaton is assigned to a disabled object it starts when the stimulus is enabled. The behaviour at the end of the animation depends on the setting of the "final action" (see section C.1 on page 9) at the time the animations ends. The default is that the animation will be deassigned from the stimulus. If a stimulus with an active animation is disabled the animation will stop and be resumed as soon as the stimulus is enabled again.

Final Actions: There are various actions that can be executed when an animation completes (see section C.1 on page 9). One of them is "signal an event". Use the `OpenEventA` function with the event name `StimServerAnimationDone` to get a handle to the event. You can then use `WaitForSingleObject` to wait for the event to be signaled or to check it's current state.

3.1 General Motion Paths

With this animation an arbitrary motion path can be read from a binary file. In Matlab you can create files by calling `NumericBinaryFile` with a `single(2,n)` array. The array has to contain pixel coordinates for each frame.

3.2 Straight Line Segment Motion Paths

With this animation the assigned object is moved along a polygon defined through its vertices. The speed of the motion is constant along the entire polygon and is specified at animation creation.

A General Commands

`[00 0]` delete all stimuli (except photo diode signal)

`[00 0 e]` enable (`e=1`) or disable (`e=0`) photo diode signal. On server startup the signal is enabled and black.

`[00 0 0 e]` enable (`e=1`) or disable (`e=0`) all stimuli (except photo diode signal)

`[00 0 1 p]` protect (`p=1`) or unprotect (`p=0`) all stimuli (except photo diode signal). A protected stimulus is unaffected by "delete all", "enable all" and "disable all" commands. The photo diode signal is always protected.

`[00 0 r g b]` set screen background. r, g, b = $0 \dots 255$ (unsigned)

`[00 1 0]` end deferred mode (and apply all changes)

`[00 1 1]` start deferred mode

`[00 1 2]` query performance counter (read 8 bytes as an uint64 after issuing this command)

`[00 1 3 a]` set the default "final action" for animations. Newly created animations will inherit this setting. Refer to section C.1 on page 9 for details.

`[00 1 4]` query and reset the global error state of the server (read the error mask as an uint16 after issuing this command). A value of 0 for the error mask indicates that no errors were detected by the server. The respective bits of the mask indicate:

1	general server errors occured – query the error code for detail (see below)
2	stimulus errors occured – query the stimuli for detailed information
4	animation errors occured – query animations for detailed information

[00 1 5 r g b α] set the default draw color. All color components are uint8 $(0 \ldots 255)$. Newly created particle, symbol or rectangular stimuli will inherit this setting. The initial color is opaque white.

[00 1 7] query and reset the error code for the most recent general error of the server (read the error state as an uint16 after issuing this command). A value of 0 for the error state indicates that no general server errors were detected.

[00 16 0] turn photo diode signal off (black)

[00 16 1] turn photo diode signal on (white)

[00 16 2] toggle photo diode signal

[00 16 3] turn on photo diode signal flicker mode (toggles with each frame)

A.1 Stimulus Creation

After sending a stimulus creation command you have to read 2 bytes (as a 16 bit unsigned integer) from the pipe. This number will be 0 if an error occured during stimulus creation. Otherwise the key (kk) of the newly created stimulus will be returned.

These commands are not deferrable.

[00 2 filename]   create a picture stimulus from the specified image file.

[00 3 kk filename]   create or replace with a picture stimulus from the specified image file.

[00 4 filename]   create a pixel shader stimulus from the specified HLSL file.

[00 5 kk filename]   create or replace with a pixel shader stimulus from the specified HLSL file.

[00 8 w h filename]   create a particle stimulus (size w×h pixels) from the specified file. w and h are uint16. For example Matlab code to create a coordinate file see section 2.1 on page 2. *Note: The ability to set w and h independently is intended for future expansions to rectangular stimuli. The current implementation is designed for stimuli with a square layout, i.e. w and h should be equal.*

[00 9 w h kk filename]   create or replace with a particle stimulus from the specified file.

[00 10]   create a pixel stimulus.

[00 11 kk]   create or replace with a pixel stimulus.

[OO 12 t s]   create a symbol stimulus of type t (uint8) and size s (uint16). The size specifies the width and the height of the symbol.

t	symbol type
1	filled circle
2	outlined circle

[OO 13 t s kk]   create or replace symbol stimulus of type t and size s.

[OO 14 filename]   create a bitmap brush from the specified image file.

[OO 15 kk filename]   create or replace with a bitmap brush from the specified image file.

[OO 18 kk filename]   create a pixel shader stimulus from the specified HLSL file. The shader is expected to operate on an existing picture object identified through the key kk. See section 2.3 on page 3 for details.

[OO 20]   create a rectangle (bar) stimulus. The initial color is inherited from the "default draw color".

A.2 Animation Creation

After sending an animation creation command you have to read 2 bytes (as a 16 bit unsigned integer) from the pipe. This number will be 0 if an error occured during animation creation. Otherwise the key (ka) of the newly created animation will be returned.

These commands are not deferrable.

[OO 130 filename]   create a motion path from the specified binary file. In Matlab you can create files by calling NumericBinaryFile with a single(2,n) array. The array should contain pixel coordinates for each frame.

[OO 132 vv]   create a straight line segment motion path object. vv is the velocity in pixels per second (16 bit unsigned integer). *A frame rate of 120 Hz is assumed.* For information on how to specify the path and start the animation see C.2 on page 9.

B   Stimulus Commands

B.1   General Stimulus Commands

[kk 0] remove (destroy) stimulus (not deferrable)

[kk 0 e] enable (e=1) or disable (e=0) stimulus

[kk 3 p] protect (p=1) or unprotect (p=0) stimulus (see section A on page 4)

[kk 3 x y] move (center of) stimulus to x, y (single floats)

[kk 7] query and reset error state of stimulus (not deferrable). You have to read the error code as an uint16 after issuing this command. A value of 0 for the error state indicates that no errors were detected for this stimulus.

[kk 8] query the position of the stimulus (not deferrable). You have to read the position as two single floats after issuing this command. With Matlab you can read an uint64 and then typecast to 'single'.

This command is particularly useful while an animation is running. It will reflect the actual position of the stimulus (independent of "deferred mode").

B.2   Picture Commands

[kk 1 α] set global alpha value for picture ($\alpha = 0 \ldots 255$, uint8)

[kk 2 i] set increment value for picture rotation to i ($i = -128 \ldots 127$)

[kk 4 φ] set orientation of picture. φ is a single float specifying the angle in degrees.

B.3   Pixel Shader Commands

[kk 1 i value] set value (single float) for parameter i of shader

[kk 2 value] set animation increment value (single float) for shader (speed)

[kk 5 i r g b α] set components ($0 \ldots 255$) for color i of shader ($i = 1 \ldots 4$)

[kk 6 value] set animated value (single float) for shader

[kk 9 width height] set stimulus size for shader. width and height are uint16.

B.4   Particle Stimulus Commands

[kk 1 1 value] set particle diameter (uint16 — initial value is 4)

[kk 1 2 value] set patch radius ($0 \ldots 1.42$ single float) for particle stimulus. A value of 0 switches back to the unpatched (rectangular) mode.

[kk 1 3 value] set Gaussian patch (single float) for particle stimulus. A value of 0 disables the Gaussian patch. For details see section 2.1.1 on page 2.

[kk 2 velocity] set velocity (single float) for particles. The velocity is expressed as the (normalized) coordinate increment per frame.

[kk 4 angle] set angle (degrees as single float) of particle movement. If each particle has it's own angle (see section 2.1 on page 2) this value is added to each of them. *In general the position of a particle is not preserved when the angle is changed.*

[kk 5 r g b α] set components (uint8, $0\dots 255$) for color of particles. (The initial color is inherited from the "default draw color".)

[kk 6 value] set animated shift value (single float).

B.5   Symbol Commands

[kk 1 1 value] set symbol size (uint16)

[kk 5 r g b α] set components (uint8, $0\dots 255$) for color of symbol. (The initial color is inherited from the "default draw color".)

This command is not yet implemented for outlined circles.

B.6   Bitmap Brush Commands

[kk 10] remove (destroy) opacity mask (not deferrable)

[kk 10 e] enable (e=1) or disable (e=0) opacity mask

[kk 11 filename] load an opacity mask for the bitmap brush from the specified image file.

[kk 13 x y] move opacity mask relative to bitmap brush (x and y are single precision floats)

B.7   Pixel Shaded Picture Commands

[kk 1 i value] set value (single float) for parameter i of shader

[kk 2 value] set animation increment value (single float) for shader (speed)

[kk 6 value] set animated value (single float) for shader

B.8   Rectangle Commands

[kk 1 1 width height] set rectangle size (uint16s). Initial size is 11×21 pixels.

[kk 4 φ] set orientation of rectangle. φ is a single float specifying the angle in degrees.

[kk 5 r g b α] set components (uint8, $0\dots 255$) for color of rectangle. (The initial color is inherited from the "default draw color".)

C Animation Commands

C.1 General Animation Commands

[ka 0] remove (destroy) animation (not deferrable)

[ka 0 a] define the terminal action for the animation. a is a uint8 bit mask. The bits have the following meaning:

1	disable the assigned stimulus
2	
4	toggle the photodiode signal[1]
8	signal an event (see section 3 on page 4)
16	restart animation (cyclic execution)
32	reverse animation[2]
64	goto initial state[2]
128	end deferred mode

Setting a to 0 restores the default behaviour of deassigning the animation from the stimulus.

Newly created animations inherit the "default final action mask" (see page 4)

Do not use "toggle the photodiode signal" together with "end deferred mode". Instead use "end deferred mode" only and toggle the photodiode while in deferred mode".

For unknown reasons some deferred commands (e.g. change symbol size) do not work as expected with "end deferred mode".

[ka 0 e kk] assign (e=1) or deassign (e=0) animation ka to/from stimulus kk

[ka 7] query and reset error state of animation (not deferrable). You have to read the error code as an uint16 after issuing this command. A value of 0 for the error state indicates that no errors were detected for this animation.

C.2 Straight Line Segments Animation Commands

[ka 11 vertices] set the vertex coordinates of the straight line segment motion path. vertices are 16 bit signed integers in pixel space. The animation starts instantly if it's already assigned to a stimulus – else it starts once it is assigned. *The maximum number of vertices to be specified with this command is 31.*

D Error Codes

D.1 General Error Codes

1 Error creating stimulus (see log window for details).

---

[1] irrespective of "deferred mode"

[2] Not implemented yet.

2 Command for non existing stimulus object (see log window for details).

3 Trying to assign non existing picture object to pixel shader (see log window for details).

4 Compilation of a shader file failed – the stimulus is not created (see log window for details).

5 Trying to create a symbol of size 0 – the stimulus is not created.

6 Trying to create a particle stimulus with width or height equal to 0 – the stimulus is not created.

D.2   Stimulus Error Codes

1 Error reading Opacity Mask file for Bitmap Brush stimulus.

2 Command length error. The command was ignored. See log window for details.

3 Invalid command error. The command was ignored. See log window for details.

4 Trying to set particle or symbol size to 0. The command was ignored.

E   Pixel Shaders

In StimServer versions prior to version 1.2 the size of the stimulus had to be specified in the shader source code. Also these global variables (width and height) had to be used within the shader in a non trivial way. Otherwise the compiler would optimze them out. Example for a simple grating:

```
extern float width  = 166.0f;
extern float height = 166.0f;

cbuffer PS_ANIMATION : register (b1)
{
    float  phase : packoffset(c0.x);
};

cbuffer PS_CONSTANT_BUFFER : register (b0)
{
    float2 center : packoffset(c0.x);
    float  orientation : packoffset(c0.z); // param 1
    float  cycle       : packoffset(c0.w); // param 2
    float4 dummy1      : packoffset(c1.x); // params 3-6
    float4 dummy2      : packoffset(c2.x); // params 7-10
};

float4 PSmain( float4 Pos : SV_POSITION ) : SV_Target
{
    if (distance(center, Pos.xy) > 83.0f) discard;
```

```
    static const float pi = 3.141592654;

    float2 p;
    sincos(radians(orientation), p.y, p.x);
    float bright = width-height +
        (sin( phase + dot(center-Pos.xy,p)*(2.0f*pi)/(cycle ? cycle : 14.0f)) + 1.0f)/2.0f;
    return float4( bright, bright, bright, 1.0f );
}
```

Begining with version 1.2 of the StimServer the size of the stimulus can be set with a separate command and is then provided to the shader through the PS_CONSTANT_BUFFER. The shader above could then look like this:

```
cbuffer PS_ANIMATION : register (b1)
{
    float  phase : packoffset(c0.x);
};

cbuffer PS_CONSTANT_BUFFER : register (b0)
{
    float2 center : packoffset(c0.x);
    float  width  : packoffset(c0.z);
    float  height : packoffset(c0.w);
    float  orientation : packoffset(c1.x); // param 1
    float  cycle  : packoffset(c1.y); // param  2
    float2 dummy1 : packoffset(c1.z); // params 3-4
    float4 dummy2 : packoffset(c2.x); // params 5-8
};

float4 PSmain( float4 Pos : SV_POSITION ) : SV_Target
{
    if (distance(center, Pos.xy) > width/2.0f) discard;
    static const float pi = 3.141592654;

    float2 p;
    sincos(radians(orientation), p.y, p.x);
    float bright =
        (sin( phase + dot(center-Pos.xy,p)*(2.0f*pi)/(cycle ? cycle : 14.0f)) + 1.0f)/2.0f;
    return float4( bright, bright, bright, 1.0f );
}
```