SIEGE Documentation

Release 0.1.9

SIEGE Development Team

March 10, 2011

CONTENTS

CHAPTER

ONE

AI

1.1 A-Star Search

SIEGE includes a generic implementation of the A* search algorithm.

It does not work on grids or any other form of data, but merely on abstract "nodes". This means that the algorithm is completely generic, and can be used for any kind of searches. Grid-based and other forms of pathfinding are created by wrapping this algorithm in their respective representations.

float **(*SGAStarScore)** (SGAStarNode* *a*, SGAstarNode* *b*)
This type of a function is used for calculating the costs and the heuristic. Note that, for the algorithm to operate properly, the heuristic *must* be optimistic - that means that it should be lower or equal to the actual cost of the path between the nodes.

Note: If a heuristic cannot be calculated, or if it would be impractical to do so, the function should return 0 for a heuristic. For more info, see also `SGAStarNode.score.h`.

Note: When the function is used for calculating a heuristic, either a or b are usually the goal.

SGbool **(*SGAStarIsGoal)** (SGAStarNode* *a*, SGAStarNode* *b*)
This type of a function should return SG_TRUE if b is a goal when coming from a, and SG_FALSE otherwise.

struct **SGAStarNode**
Contains a single, generic, A* node.

SGAStarNode* **from**
Pointer towards the start node from the current node. It is used in backtracking in order to find a path from the goal to the start.

SGList* **links**
A list of nodes which this node leads to.

void* **data**
User-provided node data. This is treaded as an opaque pointer by SIEGE.

struct **score**

float **score.g**
Contains the current cost so far, from start to the current node.

float **score.h**
Contains the heuristic - the estimated cost from the current node to the goal.

float **score.f**
Contains the sum of g and h.

struct **SGAStar**

struct **set**

SGList* **set.open**

SGList* **set.closed**

SGList* **path**

SGAStarNode* **current**

SGAStarNode* **goal**

SGbool **gfound**

struct **cb**

SGAstarScore **cb.g**

SGAstarScore **cb.h**

SGAstarIsGoal **cb.isgoal**

SGAStar* **sgAStarCreate** (SGAStarNode* *start*, SGAStarNode* *goal*, SGAStarScore *g*, SGAStarScore *h*, SGAStarIsGoal *isgoal*)

void **sgAStarDestroy** (SGAStar* *search*)

1.1.1 Searching

SGbool **sgAStarStep** (SGAStar* *search*)

SGbool **sgAStarGoalFound** (SGAStar* *search*)

SGList* **sgAStarPath** (SGAStar* *search*, SGuint* *pathlen*)

1.1.2 Node-based operations

SGAStarNode* **sgAStarNodeCreate** (void* *data*)

void **sgAStarNodeDestroy** (SGAStarNode* *node*)

void **sgAStarNodeLink** (SGAStarNode* *from*, SGAStarNode* *to*)

void **sgAStarNodeDLink** (SGAStarNode* *one*, SGAStarNode* *two*)

void **sgAStarNodeUnlink** (SGAStarNode* *from*, SGAStarNode* *to*)

void **sgAStarNodeDUnlink** (SGAStarNode* *one*, SGAStarNode* *two*)

1.2 Navigation Grid

SG_NAVGRID_CLEAR
SG_NAVGRID_WALL

SG_NAVGRID_START
SG_NAVGRID_GOAL

struct **SGNavGridData**

SGuint **x**

SGuint **y**

SGenum **type**

float **cost**

struct **SGNavGrid**

SGAstar* **search**

SGAStarNode*** **grid**

SGuint **width**

SGuint **height**

SGList* **path**

SGAStarNode* **start**

SGAStarNode* **goal**

SGbool **diag**

SGbool **wdiag**

SGNavGrid* **sgNavGridCreate**(SGuint *width*, SGuint *height*, SGbool *diag*, SGbool *wdiag*)

void **sgNavGridDestroy**(SGNavGrid* *grid*)

1.2.1 Fetching info

SGAStarNode* **sgNavGridGetNode**(SGNavGrid* *grid*, SGuint *x*, SGuint *y*)

SGNavGridData* **sgNavGridGetData**(SGNavGrid* *grid*, SGuint *x*, SGuint *y*)

1.2.2 Modifying grid

void **sgNavGridAddClear**(SGNavGrid* *grid*, SGuint *x*, SGuint *y*)

void **sgNavGridAddWall**(SGNavGrid* *grid*, SGuint *x*, SGuint *y*)

void **sgNavGridAddStart**(SGNavGrid* *grid*, SGuint *x*, SGuint *y*)

void **sgNavGridAddGoal**(SGNavGrid* *grid*, SGuint *x*, SGuint *y*)

1.2.3 Searching

void **sgNavGridSearchCreate**(SGNavGrid* *grid*)

SGbool **sgNavGridSearchStep**(SGNavGrid* *grid*)

SGbool **sgNavGridGoalFound**(SGNavGrid* *grid*)

SGList* **sgNavGridSearchPath** (SGNavGrid* *grid*, SGuint* *pathlen*)

CHAPTER

TWO

AUDIO

2.1 Buffer

struct **SGAudioBuffer**

The buffer holds the audio data for the SGAudioSource "sources" to play. Please note that a single buffer can be used in multiple sources.

void* **handle**

Internal handle.

> **Warning:** For internal use only.

SGAudioBuffer* **sgAudioBufferCreate**(const char* *fname*)

Create an audio buffer.

Parameters

- **fname** – Filename of the audio file to load.

Returns The newly created buffer if successful, NULL otherwise.

void **sgAudioBufferDestroy**(SGAudioBuffer* *buffer*)

Destroy an audio buffer.

Note: It should not be used anymore after this call.

Parameters

- **buffer** – The buffer to destroy.

void **sgAudioBufferSetData**(SGAudioBuffer* *buffer*, SGuint *channels*, SGuint *format*, SGuint *frequency*, void* *data*, SGuint *datalen*)

Set the data of an audio buffer. The function creates a copy, so the data may be freed after the call.

Parameters

- **buffer** – The buffer to modify.
- **channels** – Number of channels in the data.
- **format** – Audio format.
- **frequency** – Frequency (sample rate) of the data.
- **data** – Audio data to pass to the buffer.

- **datalen** – Length of the audio data in bytes.

2.2 Source

struct **SGAudioSourceDispatch**
Audio source usage handle - the dispatch helps handle the sources. Each dispatch refers to a backend source, and is (when in use) assigned a frontend source.

This makes sure that sources are managed in a conservative fashion and recycled whenever possible.

SGAudioSource* **source**
Currently assigned source to the dispatch.

> **Warning:** For internal use only.

void* **handle**
Internal handle.

> **Warning:** For internal use only.

struct **SGAudioSource**
This is the actual object responsible for positioning and playing an audio source.

SGAudioSourceDispatch* **SGAudioSource.dispatch**
Pointer to the dispatch this source is currently assigned to.

float **SGAudioSource.priority**
Priority of the source.

If no more sources can be created in the backend for play, the higher priority sources will have precedence over lower priority sources.

That way you can, for example, make the explosion near you "override" a bird tweeting on a tree.

SGAudioSource* **sgAudioSourceCreate**(float *priority*, float *volume*, float *pitch*, SGbool *looping*)
Create an audio source.

Parameters

- **priority** – Source priority.
- **volume** – Volume of the source, from 0.0 to 1.0.
- **pitch** – Pitch of the source, 1.0 being no adjustment.
- **looping** – Should the source loop?

Returns The newly created source if successful, NULL otherwise.

void **sgAudioSourceDestroy**(SGAudioSource* *source*)
Destroy an audio source.

2.2.1 Basic functions

These control the basic source functionality - playing, pausing, stopping, etc...

void **sgAudioSourcePlay**(SGAudioSource* *source*)

SGbool **sgAudioSourceIsPlaying**(SGAudioSource* *source*)
Playing a source.

void **sgAudioSourcePause**(SGAudioSource* *source*)
SGbool **sgAudioSourceIsPaused**(SGAudioSource* *source*)
Pausing a source.

void **sgAudioSourceRewind**(SGAudioSource* *source*)
Rewinding an audio source.

void **sgAudioSourceStop**(SGAudioSource* *source*)
SGbool **sgAudioSourceIsStopped**(SGAudioSource* *source*)
Stopping an audio source.

SGbool **sgAudioSourceIsActive**(SGAudioSource* *source*)
Is the source active (in use)?

2.2.2 Queueing buffers

Queueing buffers (as opposed to simply assigning a single buffer to a source) can be used for things like smooth streaming of large audio data such as music.

void **sgAudioSourceQueueBuffers**(SGAudioSource* *source*, SGAudioBuffer** *buffers*, SGuint *numbuffers*)
void **sgAudioSourceQueueBuffer**(SGAudioSource* *source*, SGAudioBuffer* *buffer*)
Queue buffers into a source. The latter function is semantically equivalent to calling the former with a single buffer.

2.2.3 Position

void **sgAudioSourceSetPosition3f**(SGAudioSource* *source*, float *x*, float *y*, float *z*)
void **sgAudioSourceSetPosition3f**(SGAudioSource* *source*, float* *x*, float* *y*, float* *z*)
Set/get the x,y,z position.

void **sgAudioSourceSetPosition2f**(SGAudioSource* *source*, float *x*, float *y*)
void **sgAudioSourceSetPosition2f**(SGAudioSource* *source*, float* *x*, float* *y*)
Set/get the x,y position, and z implicitly to 0.

2.2.4 Velocity

void **sgAudioSourceSetVelocity3f**(SGAudioSource* *source*, float *x*, float *y*, float *z*)
void **sgAudioSourceSetVelocity3f**(SGAudioSource* *source*, float* *x*, float* *y*, float* *z*)
Set/get the x,y,z velocity.

void **sgAudioSourceSetVelocity2f**(SGAudioSource* *source*, float *x*, float *y*)
void **sgAudioSourceSetVelocity2f**(SGAudioSource* *source*, float* *x*, float* *y*)
Set/get the x,y velocity, and z implicitly to 0.

2.2.5 Pitch

void **sgAudioSourceSetPitch**(SGAudioSource* *source*, float *pitch*)
float **sgAudioSourceGetPitch**(SGAudioSource* *source*)
Set/get the source pitch.

2.2.6 Volume

void **sgAudioSourceSetVolume**(SGAudioSource* *source*, float *volume*)
float **sgAudioSourceGetVolume**(SGAudioSource* *source*)
Set/get the source volume.

2.2.7 Looping

void **sgAudioSourceSetLooping**(SGAudioSource* *source*, SGbool *looping*)
SGbool **sgAudioSourceGetLooping**(SGAudioSource* *source*)
Set or query whether the source loops.

CHAPTER

THREE

MISCELLANEOUS

SIEGE_TEST
Use test functionality. Deprecated since version 0.1.0: This is included for testing purposes and therefore may be removed in the future. Defined to indicate that some "test" functionality should be used in the modules.

SG_EXPORT
This is used for all SIEGE functions. SIEGE currently defaults to the the cdecl calling convention.

SG_MAX(x, y)
SG_MIN(x, y)
SG_ABS(x)

struct **SGModuleInfo**

SGushort **vmajor**
SGushort **vminor**
SGushort **vpatch**
SIEGE version.

SGushort **mmajor**
SGushort **mminor**
SGushort **mpatch**
Module version.

SGuint **type**
Module type.

char* **name**
Module name.

void* **data**
Module data (treated as opaque by SIEGE).

3.1 Types

SGvoid
SGbool
SGenum

SGchar
SGwchar
SGdchar
Character types.

SGbyte
SGubyte
8-bit integer.

SGshort
SGushort
16-bit integer.

SGint
SGuint
32-bit integer.

SGlong
SGulong
64-bit integer.

SGfloat
SGdouble
Floating-point.

3.2 Hints

Hints are used to give the compile more info, enabling it to produce warning messages for deprecation, invalid printf-like syntax or similar.

SG_HINT_DEPRECATED
Indicates that the item is deprecated and thus scheduled for removal. Use of deprecated functions/types/constants/... is not recommended.

SG_HINT_PRINTF (str, chk)
Indicates that the syntax of the function resembles that of printf.

Parameters

- **str** – Index of the format string (starting with 1)
- **chk** – Index of the first vararg (starting with 1) or 0 if the function accepts va_args directly

3.3 Version information

SG_VERSION_MAJOR
SG_VERSION_MINOR
SG_VERSION_PATCH

SG_VERSION_STRING

3.4 Booleans

SG_TRUE
SG_FALSE
These are typically used in conjunction with `SGbool`.

3.5 Floating-point constants

SG_NAN
Used by some functions to indicate an "invalid value" when returning.

SG_INF
Currently used with physics, to indicate infinite mass (or moment of inertia).

3.6 Module error values

SG_OK
SG_NO_ERROR
SG_UNKNOWN_ERROR
SG_INVALID_VALUE
These values are to be returned from backend modules, to indicate roughly if (and which) error has occured.

They are currently not used in the frontend -> client space, but they may be in the future.

3.7 Module types

SG_MODULE_WINDOW
SG_MODULE_INPUT
SG_MODULE_CORE
SG_MODULE_GRAPHICS
SG_MODULE_GRAPHICSLOAD
SG_MODULE_AUDIO
SG_MODULE_AUDIOLOAD
SG_MODULE_FONTLOAD
SG_MODULE_PHYSICS
Used in modules, to indicate which interface groups they implement. These are *not* used in SIEGE frontend - they are merely used in other modules, for checking modules compatibility.

3.8 Audio formats

SG_AUDIO_FORMAT_S8
SG_AUDIO_FORMAT_S16
SG_AUDIO_FORMAT_S24
SG_AUDIO_FORMAT_S32

SG_AUDIO_FORMAT_U8
SG_AUDIO_FORMAT_U16
SG_AUDIO_FORMAT_U24
SG_AUDIO_FORMAT_U32

SG_AUDIO_FORMAT_F
SG_AUDIO_FORMAT_D

3.9 Graphics primitive types

SG_GRAPHICS_PRIMITIVE_POINTS

SG_GRAPHICS_PRIMITIVE_LINES
SG_GRAPHICS_PRIMITIVE_LINE_STRIP
SG_GRAPHICS_PRIMITIVE_LINE_FAN
SG_GRAPHICS_PRIMITIVE_LINE_LOOP

SG_GRAPHICS_PRIMITIVE_TRIANGLES
SG_GRAPHICS_PRIMITIVE_TRIANGLE_STRIP
SG_GRAPHICS_PRIMITIVE_TRIANGLE_FAN

SG_GRAPHICS_PRIMITIVE_QUADS
SG_GRAPHICS_PRIMITIVE_QUAD_STRIP

SG_GRAPHICS_PRIMITIVE_CONVEX_POLYGON
SG_GRAPHICS_PRIMITIVE_CONCAVE_POLYGON
SG_GRAPHICS_PRIMITIVE_INTERSECTING_POLYGON

3.10 Graphics info flags

Deprecated since version 0.1.0: These may be removed in the future.

SG_GRAPHICS_TEXTURE_NPOT
: This flag indicates that the backend does not support non-power-of-two textures and that conversion to such sizes may be required by the frontend.

3.11 Physics shapes

SG_PHYSICS_SHAPE_SEGMENT
SG_PHYSICS_SHAPE_POLYGON
SG_PHYSICS_SHAPE_CIRCLE

3.12 Physics body types

SG_PHYSICS_BODY_PASSIVE
SG_PHYSICS_BODY_NORMAL
SG_PHYSICS_BODY_STATIC

SG_PHYSICS_BODY_SEMISTATIC
: A body type with seemingly infinite mass, but still movable (usually through user interaction).

Note: May be deprecated in the future.

3.13 Window flags

SG_WINDOW_FULLSCREEN
SG_WINDOW_RESIZABLE

3.14 Keyboard keys

SG_KEYBOARD_KEY_UNKNOWN
SG_KEYBOARD_KEY_SPACE
SG_KEYBOARD_KEY_ESC
SG_KEYBOARD_KEY_F1
SG_KEYBOARD_KEY_F2
SG_KEYBOARD_KEY_F3
SG_KEYBOARD_KEY_F4
SG_KEYBOARD_KEY_F5
SG_KEYBOARD_KEY_F6
SG_KEYBOARD_KEY_F7
SG_KEYBOARD_KEY_F8
SG_KEYBOARD_KEY_F9
SG_KEYBOARD_KEY_F10
SG_KEYBOARD_KEY_F11
SG_KEYBOARD_KEY_F12
SG_KEYBOARD_KEY_F13
SG_KEYBOARD_KEY_F14
SG_KEYBOARD_KEY_F15
SG_KEYBOARD_KEY_F16
SG_KEYBOARD_KEY_F17
SG_KEYBOARD_KEY_F18
SG_KEYBOARD_KEY_F19
SG_KEYBOARD_KEY_F20
SG_KEYBOARD_KEY_F21
SG_KEYBOARD_KEY_F22
SG_KEYBOARD_KEY_F23
SG_KEYBOARD_KEY_F24
SG_KEYBOARD_KEY_F25
SG_KEYBOARD_KEY_UP
SG_KEYBOARD_KEY_DOWN
SG_KEYBOARD_KEY_LEFT
SG_KEYBOARD_KEY_RIGHT
SG_KEYBOARD_KEY_LSHIFT
SG_KEYBOARD_KEY_RSHIFT
SG_KEYBOARD_KEY_LCTRL
SG_KEYBOARD_KEY_RCTRL
SG_KEYBOARD_KEY_LALT
SG_KEYBOARD_KEY_RALT
SG_KEYBOARD_KEY_TAB
SG_KEYBOARD_KEY_ENTER
SG_KEYBOARD_KEY_BACKSPACE
SG_KEYBOARD_KEY_INSERT
SG_KEYBOARD_KEY_DELETE
SG_KEYBOARD_KEY_HOME
SG_KEYBOARD_KEY_END
SG_KEYBOARD_KEY_PAGEUP
SG_KEYBOARD_KEY_PAGEDOWN
SG_KEYBOARD_KEY_KP0
SG_KEYBOARD_KEY_KP1
SG_KEYBOARD_KEY_KP2
SG_KEYBOARD_KEY_KP3

SG_KEYBOARD_KEY_KP4
SG_KEYBOARD_KEY_KP5
SG_KEYBOARD_KEY_KP6
SG_KEYBOARD_KEY_KP7
SG_KEYBOARD_KEY_KP8
SG_KEYBOARD_KEY_KP9
SG_KEYBOARD_KEY_KP_ADD
SG_KEYBOARD_KEY_KP_SUBTRACT
SG_KEYBOARD_KEY_KP_MULTIPLY
SG_KEYBOARD_KEY_KP_DIVIDE
SG_KEYBOARD_KEY_KP_DECIMAL
SG_KEYBOARD_KEY_KP_EQUAL
SG_KEYBOARD_KEY_KP_ENTER

3.15 Mouse buttons

SG_MOUSE_BUTTON_LEFT
SG_MOUSE_BUTTON_RIGHT
SG_MOUSE_BUTTON_MIDDLE

INDEX

S