jubilant-funicular

Generated by Doxygen 1.8.13

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 nta::AudioManager Class Reference

Static Public Member Functions

- static void init ()

  initializes SDL_Mixer
- static void destroy ()

  frees all music
- static SoundEffect ∗ getSoundEffect (crstring effectPath)

  returns sound
- static Music ∗ **getMusic** (crstring musicPath)

3.1.1 Detailed Description

Definition at line 65 of file AudioManager.h.

The documentation for this class was generated from the following files:

- include/nta/AudioManager.h
- src/AudioManager.cpp

3.2 nta::Camera2D Class Reference

represents a camera in two dimensions from which the world is viewed

```
#include <Camera2D.h>
```

Public Member Functions

- Camera2D ()
    constructors
- **Camera2D** (crvec2 center)
- **Camera2D** (crvec2 center, crvec2 dimensions)
- ~Camera2D ()
    destructor
- glm::mat3 getCameraMatrix () const
    returns the 3x3 matrix representing the camera's view
- glm::vec4 getBoundsCenter () const
    returns camera bounds in the given format
- glm::vec4 **getBoundsTopLeft** () const
- glm::vec2 getCenter () const
    returns the center, top left coordinate, and dimensions of the camera's view
- glm::vec2 **getTopLeft** () const
- glm::vec2 **getDimensions** () const
- glm::vec2 mouseToGame (crvec2 mouse, crvec2 windowDimensions) const
    converts mouse coordinates to world coordinates
- void setCenter (crvec2 center)
    sets the values of the camera's fields
- void **setCenter** (float x, float y)
- void **setDimensions** (crvec2 dimensions)
- void **setDimensions** (float w, float h)
- void translateCenter (crvec2 translation)
    moves the camera around the world
- void **translateCenter** (float dx, float dy)
- void scaleDimensions (crvec2 dilation)
    scales the camera's field of view
- void **scaleDimensions** (float dw, float dh)

3.2.1 Detailed Description

represents a camera in two dimensions from which the world is viewed

Definition at line 8 of file Camera2D.h.

3.2.2 Member Function Documentation

3.2.2.1 mouseToGame()

```
glm::vec2 nta::Camera2D::mouseToGame (
            crvec2 mouse,
            crvec2 windowDimensions ) const
```

converts mouse coordinates to world coordinates

[a,b]->[0,b-a]->[0,d-c]->[c,d]

Definition at line 48 of file Camera2D.cpp.

The documentation for this class was generated from the following files:

- include/nta/Camera2D.h
- src/Camera2D.cpp

3.3 nta::CharGlyph Struct Reference

represents a single char in the texture

`#include <SpriteFont.h>`

Public Attributes

- glm::vec4 uvRect
  the rectangle containing this glyph in the texture
- glm::vec2 size
  the size of the rendered glyph

3.3.1 Detailed Description

represents a single char in the texture

Definition at line 16 of file SpriteFont.h.

3.3.2 Member Data Documentation

3.3.2.1 size

`glm::vec2 nta::CharGlyph::size`

the size of the rendered glyph

Definition at line 20 of file SpriteFont.h.

Referenced by nta::SpriteFont::drawText(), and nta::SpriteFont::measure().

3.3.2.2 uvRect

`glm::vec4 nta::CharGlyph::uvRect`

the rectangle containing this glyph in the texture

Definition at line 18 of file SpriteFont.h.

Referenced by nta::SpriteFont::drawText().

The documentation for this struct was generated from the following file:

- include/nta/SpriteFont.h

3.4 nta::Compressor Class Reference

Static class for compressing byte buffers.

```
#include <Compressor.h>
```

Static Public Member Functions

- static std::vector< GLubyte > decompress (const std::vector< GLubyte > &data)

    decompressed data that was compressed by this class
- static std::vector< GLubyte > compress (const std::vector< GLubyte > &data)

    compresses a bye buffer

3.4.1 Detailed Description

Static class for compressing byte buffers.

Definition at line 53 of file Compressor.h.

The documentation for this class was generated from the following files:

- include/nta/Compressor.h
- src/Compressor.cpp

3.5 nta::FontMap Class Reference

represents the organization of a texture containing the characters

```
#include <SpriteFont.h>
```

Public Member Functions

- FontMap ()

    constructor and destructor
- glm::vec2 getBoundingDimensions () const

    returns the dimensions of the rectangle that contains the FontMap
- void addRect (char c, crvec2 dimensions)

    adds a rectangle and associates it with c (replacing any preexisting rectangle)
- void position ()

    positions map so that the topleft is at (0,0)

Public Attributes

- friend **SpriteFont**

3.5.1 Detailed Description

represents the organization of a texture containing the characters

Definition at line 23 of file SpriteFont.h.

The documentation for this class was generated from the following files:

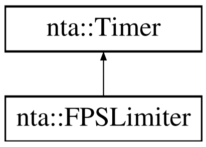
- include/nta/SpriteFont.h
- src/FontMap.cpp

3.6 nta::FPSLimiter Class Reference

used to cap the fps of the program at a specific value

```
#include <FPSLimiter.h>
```

Inheritance diagram for nta::FPSLimiter:



Public Member Functions

- FPSLimiter ()

  constructor and destructor
- void setMaxFPS (float maxFPS)

  sets maximum allowed fps
- float getFPS () const

  gets most recently calculated fps
- long double end ()

  ends fps calculations, delaying if necessary to cap fps

Additional Inherited Members

3.6.1 Detailed Description

used to cap the fps of the program at a specific value

Definition at line 8 of file FPSLimiter.h.

The documentation for this class was generated from the following files:

- include/nta/FPSLimiter.h
- src/FPSLimiter.cpp

3.7 nta::GLSLProgram Class Reference

represents a program written in GLSL comprised of a vertex shader and a fragment shader

```
#include <GLSLProgram.h>
```

Public Member Functions

- GLSLProgram ()

  constructor and destructor
- GLint getUniformLocation (crstring uniformName) const

  returns the location of a uniform in the shaders
- bool isLinked () const

  returns whether or not the shaders have been linked
- void addAttribute (crstring attributeName)

  makes an attribute useful and assigns it the next available location
- void linkShaders ()

  links the compiled shaders to this program
- void use () const

  binds this program
- void unuse () const

  unbinds this program

Friends

- class **SystemManager**

3.7.1 Detailed Description

represents a program written in GLSL comprised of a vertex shader and a fragment shader

Definition at line 13 of file GLSLProgram.h.

The documentation for this class was generated from the following files:

- include/nta/GLSLProgram.h
- src/GLSLProgram.cpp

3.8 nta::GLTexture Struct Reference

represents a texture

```
#include <GLTexture.h>
```

Public Attributes

- GLuint id

    the id of the texture
- GLint width

    the width and height, respectively, of the texture
- GLint **height**

3.8.1 Detailed Description

represents a texture

Definition at line 12 of file GLTexture.h.

3.8.2 Member Data Documentation

3.8.2.1 height

```
GLint nta::GLTexture::height
```

Definition at line 16 of file GLTexture.h.

3.8.2.2 id

```
GLuint nta::GLTexture::id
```

the id of the texture

Definition at line 14 of file GLTexture.h.

Referenced by nta::Sprite::render().

3.8.2.3 width

```
GLint nta::GLTexture::width
```

the width and height, respectively, of the texture

Definition at line 16 of file GLTexture.h.

The documentation for this struct was generated from the following file:

- include/nta/GLTexture.h

3.9 nta::Glyph Struct Reference

represents what is essentially a sprite

```
#include <SpriteBatch.h>
```

Public Member Functions

- **Glyph** (crvec4 posRect, crvec4 uvRect, GLuint texture, float d, crvec4 color)

Public Attributes

- GLuint textureID

  the texture used by the glyph
- float depth

  the depth of the glyph
- Vertex2D topLeft

  the vertices of the four corners of the glyph
- Vertex2D **topRight**
- Vertex2D **botRight**
- Vertex2D **botLeft**

3.9.1 Detailed Description

represents what is essentially a sprite

Definition at line 11 of file SpriteBatch.h.

The documentation for this struct was generated from the following file:

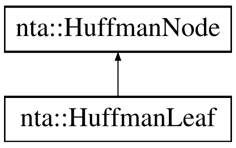
- include/nta/SpriteBatch.h

3.10 nta::HuffmanLeaf Class Reference

represents a leaf in a Huffman tree

```
#include <Compressor.h>
```

Inheritance diagram for nta::HuffmanLeaf:



Public Member Functions

- HuffmanLeaf ()

    basic constructor
- HuffmanLeaf (GLubyte data, int freq)

    constructs a leaf with given data and freq
- ∼HuffmanLeaf ()

    destroys leaf
- GLubyte getData () const

    returns m_data

Additional Inherited Members

3.10.1 Detailed Description

represents a leaf in a Huffman tree

Definition at line 38 of file Compressor.h.

The documentation for this class was generated from the following files:

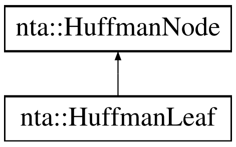
- include/nta/Compressor.h
- src/HuffmanLeaf.cpp

3.11 nta::HuffmanNode Class Reference

A node in a Huffman tree.

`#include <Compressor.h>`

Inheritance diagram for nta::HuffmanNode:



Public Member Functions

- HuffmanNode ()

    basic constructor
- HuffmanNode (HuffmanNode ∗l, HuffmanNode ∗r)

    sets l and r as children of this and sets m_freq to the sum of their frequencies
- virtual ∼HuffmanNode ()

    recursively destroys node
- auto getEncodings (crstring enc="") const -> std::map< GLubyte, std::string >

    returns map of all the bytes and how they are encoded
- HuffmanNode ∗ getLeft () const

    returns children
- HuffmanNode ∗ **getRight** () const
- bool hasChildren () const

    returns whether or not the node has children
- int getFrequency () const

    returns the frequency of the node

Protected Attributes

- int m_freq

  the frequency of the nodes associated bytes

3.11.1 Detailed Description

A node in a Huffman tree.

Definition at line 13 of file Compressor.h.

The documentation for this class was generated from the following files:

- include/nta/Compressor.h
- src/HuffmanNode.cpp

3.12 nta::ImageLoader Class Reference

loads images as GLTextures

```
#include <GLTexture.h>
```

Friends

- class **ResourceManager**

3.12.1 Detailed Description

loads images as GLTextures

Definition at line 19 of file GLTexture.h.

The documentation for this class was generated from the following files:

- include/nta/GLTexture.h
- src/GLTexture.cpp

3.13 nta::InputManager Class Reference

keeps track of all input

```
#include <InputManager.h>
```

Static Public Member Functions

- static glm::vec2 getMouseCoords ()

    returns the mouse's coordinates
- static glm::vec2 getMouseCoordsStandard (int height)

    returns the mouse's coordinates with the y axis flipped (0 represents the bottom of the screen instead of top)
- static MouseWheelMotion getMouseWheelMotion ()

    returns the mouse wheel's motion
- static bool isPressed (unsigned int key)

    returns whether or not specified key is pressed
- static bool justPressed (unsigned int key)

    returns whether or not the key was just pressed this frame
- static bool justReleased (unsigned int key)

    returns whether or not the key was just released this frame
- static void pressKey (unsigned int key)

    tells InputManager that specified key was pressed
- static void releaseKey (unsigned int key)

    tells InputManager that specified key was released
- static void setMouseCoords (float x, float y)

    tells InputManager where the mouse is
- static void setMouseWheelMotion (const MouseWheelMotion &motion)

    tells InputManager how the wheel is rolling
- static void update (SDL_Event &event)

    updates the state of m_KeyMap
- static void updatePrev ()

    updates the state of m_prevKeyMap

3.13.1 Detailed Description

keeps track of all input

Definition at line 12 of file InputManager.h.

The documentation for this class was generated from the following files:

- include/nta/InputManager.h
- src/InputManager.cpp

3.14 nta::IOManager Class Reference

Handles binary file operations.

```
#include <IOManager.h>
```

Static Public Member Functions

- static void readFileToBuffer (crstring filePath, FileBuffer &buffer)

    stores the entire contents of a file in a buffer
- static void writeFileFromBuffer (crstring filePath, const FileBuffer &buffer)

    stores the entire contents of a buffer in a file
- static void writeFloatLE (float val, std::ofstream &file)

    writes/reads a float to/from a file
- static void **writeFloatLE** (float val, FileBuffer &buffer)
- static float **readFloatLE** (std::ifstream &file)
- static float **readFloatLE** (const FileBuffer &buffer, int pos)
- static void **writeFloatBE** (float val, std::ofstream &file)
- static void **writeFloatBE** (float val, FileBuffer &buffer)
- static float **readFloatBE** (std::ifstream &file)
- static float **readFloatBE** (const FileBuffer &buffer, int pos)
- static void writeIntLE (int val, std::ofstream &file)

    writes/reads an int to/from a file
- static void **writeIntLE** (int val, FileBuffer &buffer)
- static int **readIntLE** (std::ifstream &file)
- static int **readIntLE** (const FileBuffer &buffer, int pos)
- static void **writeIntBE** (int val, std::ofstream &file)
- static void **writeIntBE** (int val, FileBuffer &buffer)
- static int **readIntBE** (std::ifstream &file)
- static int **readIntBE** (const FileBuffer &buffer, int pos)
- static void writeShortLE (short val, std::ofstream &file)

    writes/reads a short to/from a file
- static void **writeShortLE** (short val, FileBuffer &buffer)
- static short **readShortLE** (std::ifstream &file)
- static short **readShortLE** (const FileBuffer &buffer, int pos)
- static void **writeShortBE** (short val, std::ofstream &file)
- static void **writeShortBE** (short val, FileBuffer &buffer)
- static short **readShortBE** (std::ifstream &file)
- static short **readShortBE** (const FileBuffer &buffer, int pos)

3.14.1 Detailed Description

Handles binary file operations.

Definition at line 13 of file IOManager.h.

The documentation for this class was generated from the following files:

- include/nta/IOManager.h
- src/IOManager.cpp

3.15 nta::Logger Class Reference

stores program information in internal and external logs

```
#include <Logger.h>
```

Static Public Member Functions

- static void createLog ()

  creates the log
- static void writeToLog (crstring entry)

  writes an entry in the log
- static void writeErrorToLog (crstring error)

  writes entry in log and then exits program

3.15.1 Detailed Description

stores program information in internal and external logs

Definition at line 10 of file Logger.h.

The documentation for this class was generated from the following files:

- include/nta/Logger.h
- src/Logger.cpp

3.16 nta::Music Class Reference

Represents a longer piece of music.

```
#include <AudioManager.h>
```

Public Member Functions

- void play (int numLoops=1) const

  plays music
- void pause () const

  pauses music (can be resumed)
- void stop () const

  stops music (must be replayed from beginning)
- void resume () const

  resumes paused music

Friends

- class **AudioManager**

3.16.1 Detailed Description

Represents a longer piece of music.

Definition at line 33 of file AudioManager.h.

The documentation for this class was generated from the following file:

- include/nta/AudioManager.h

3.17 nta::Particle2D Struct Reference

Represents a simple 2d particle.

```
#include <ParticleBatch2D.h>
```

Public Member Functions

- **Particle2D** (crvec2 c, crvec2 v, crvec4 col)

Public Attributes

- glm::vec2 **center**
- glm::vec2 **velocity**
- glm::vec4 **color**
- float **life**

3.17.1 Detailed Description

Represents a simple 2d particle.

Definition at line 8 of file ParticleBatch2D.h.

The documentation for this struct was generated from the following file:

- include/nta/ParticleBatch2D.h

3.18 nta::ParticleBatch2D Class Reference

Represents a batch of particles of the same "type".

```
#include <ParticleBatch2D.h>
```

Public Member Functions

- ParticleBatch2D ()

  basic constructor
- ∼ParticleBatch2D ()

  deletes particles
- void init (float il, float dr, float r, int mp, int tex, std::function< void(Particle2D &, float)> updateFunc)

  initializes particle batch by specifying properties
- void addParticle (Particle2D p)

  adds a particle to the batch
- void draw (SpriteBatch &batch) const

  draws all the particles
- void update (float dt)

  updates the particles
- void clear ()

  removes all particles

3.18.1 Detailed Description

Represents a batch of particles of the same "type".

Definition at line 19 of file ParticleBatch2D.h.

The documentation for this class was generated from the following files:

- include/nta/ParticleBatch2D.h
- src/ParticleBatch2D.cpp

3.19 nta::ParticleEngine2D Class Reference

Responsible for handling multiple particle batches.

```
#include <ParticleEngine2D.h>
```

Public Member Functions

- ParticleEngine2D ()

  basic constructor
- ∼ParticleEngine2D ()

  deletes batches
- void addBatch (ParticleBatch2D ∗batch)

  adds a batch
- void draw (SpriteBatch &batch) const

  renders all batches
- void update (float dt) const

  updates all batches

3.19.1 Detailed Description

Responsible for handling multiple particle batches.

Definition at line 10 of file ParticleEngine2D.h.

The documentation for this class was generated from the following files:

- include/nta/ParticleEngine2D.h
- src/ParticleEngine2D.cpp

3.20 nta::Primitive Struct Reference

represents a primitive (point, line, triangle, etc.)

```
#include <PrimitiveBatch.h>
```

Public Member Functions

- Primitive ()

  constructors
- **Primitive** (const std::initializer_list< Vertex2D > &verts, GLuint texID, float d)
- template<class Iterator >

  Primitive (Iterator first, Iterator last, GLuint texID, float d)
- ~Primitive ()

  destructor

Public Attributes

- float depth

  the depth of the primitive
- GLuint textureID

  the texture used by the primitive
- std::vector< Vertex2D > vertices

  the vertices that make up the primitive

3.20.1 Detailed Description

represents a primitive (point, line, triangle, etc.)

Definition at line 8 of file PrimitiveBatch.h.

The documentation for this struct was generated from the following file:

- include/nta/PrimitiveBatch.h

3.21 nta::PrimitiveBatch Class Reference

represents a collection of primitives to be drawn

```
#include <PrimitiveBatch.h>
```

Public Member Functions

- PrimitiveBatch ()

  constructor and destructor
- void init ()

  initializes the batch
- void begin ()

  begins collection of primitive
- void end ()

  ends collection of primitive and prepares for rendering
- void addPrimitive (Primitive ∗primitive)

  adds a primitive to the batch
- void **addPrimitive** (const std::initializer_list< Vertex2D > &vertices, GLuint textureID, float depth=1)
- template<class Iterator >

  void **addPrimitive** (Iterator first, Iterator last, GLuint textureID, float depth=1)
- void render () const

  renders the primitives

3.21.1 Detailed Description

represents a collection of primitives to be drawn

Definition at line 32 of file PrimitiveBatch.h.

The documentation for this class was generated from the following files:

- include/nta/PrimitiveBatch.h
- src/PrimitiveBatch.cpp

3.22 nta::Random Class Reference

Used for generating random numbers.

```
#include <Random.h>
```

Static Public Member Functions

- static void init ()

  initializes random number generation
- static bool randBool ()

  randomly returns true or false
- static long randInt (long min, long max)

  returns a random int in the specified range exclusive (uniform distribution)
- static long **randInt** (long max)
- static long **randInt** ()
- static float randFloat (float min, float max)

  returns a random float in the specified range (uniform distribution)
- static float **randFloat** (float max)
- static float **randFloat** ()
- static float randGaussian (float mean, float sd)

  returns a random float using the specified distribution
- static std::default_random_engine getRNG ()

  returns the random number generator

3.22.1 Detailed Description

Used for generating random numbers.

Definition at line 12 of file Random.h.

The documentation for this class was generated from the following files:

- include/nta/Random.h
- src/Random.cpp

3.23 nta::RenderBatch Struct Reference

stores information about batches of vertices with the same texture in a vertex buffer object

```
#include <SpriteBatch.h>
```

Public Member Functions

- RenderBatch (GLuint t, GLuint o, GLuint n, GLenum m=GL_TRIANGLES)

  constructor

Public Attributes

- GLuint textureID

  the texture used by the batch
- GLuint offset

  the starting point of the batch in the vertex buffer
- GLuint numVertices

  the number of vertices comprising the vertex buffer
- GLenum mode

  the primitive type to be drawn (GL_POINTS, GL_LINES, etc.)

3.23.1 Detailed Description

stores information about batches of vertices with the same texture in a vertex buffer object

Definition at line 38 of file SpriteBatch.h.

The documentation for this struct was generated from the following file:

- include/nta/SpriteBatch.h

3.24 nta::ResourceManager Class Reference

Handles storing and retrieving textures so an image isn't loaded multiple times.

```
#include <ResourceManager.h>
```

Static Public Member Functions

- static GLTexture & getTexture (crstring imagePath, GLint minFilt=GL_LINEAR_MIPMAP_LINEAR, GLint magFilt=GL_LINEAR, crvec2 dimensions=glm::vec2(0))

  returns the resource with the given path, loading it if need be
- static GLTexture & **getTexture** (crstring imagePath, crvec2 dimensions, GLint minFilt=GL_LINEAR_MIPM↩ AP_LINEAR, GLint magFilt=GL_LINEAR)
- static SpriteFont ∗ **getSpriteFont** (crstring fontPath, int fontSize=32)
- static void deleteTexture (crstring imagePath)

  removes the resource with the given path from the map and deletes it
- static void **deleteSpriteFont** (crstring fontPath, int fontSize=32)
- static void **destroy** ()

3.24.1 Detailed Description

Handles storing and retrieving textures so an image isn't loaded multiple times.

Definition at line 9 of file ResourceManager.h.

The documentation for this class was generated from the following files:

- include/nta/ResourceManager.h
- src/ResourceManager.cpp

3.25 nta::Screen Class Reference

Represents a game screen.

```
#include <Screen.h>
```

Public Member Functions

- Screen ()

    basic constructor and destructor
- ScreenState getState () const

    returns state of screen
- virtual int getEscIndex () const

    sets/gets various screen indices
- virtual int **getXIndex** () const
- virtual int **getNextIndex** () const
- virtual int **getIndex** () const
- virtual void **setIndices** (int index, int escIndex, int xIndex)
- virtual void setWindow (crstring title)

    sets the window to associate with this screen
- virtual void render ()=0

    renders screen
- virtual void update ()=0

    updates screen
- virtual void handleInput ()

    handles user input
- virtual void onFocus ()

    called when the screen becomes active
- virtual void offFocus ()

    called when the screen is no longer active
- virtual void init ()=0

    initializes the screen

Protected Attributes

- ScreenState m_state = ScreenState::NONE

    the state of this screen
- Window $*$ m_window = nullptr

    the window the screen is rendered in
- int m_nextIndex = -1

    the index of the screen to go to in special circumstances

3.25.1 Detailed Description

Represents a game screen.

Definition at line 9 of file Screen.h.

The documentation for this class was generated from the following files:

- include/nta/Screen.h
- src/Screen.cpp

3.26 nta::ScreenManager Class Reference

Manages a collection of screens.

```
#include <ScreenManager.h>
```

Public Member Functions

- ScreenManager (crstring title, float maxFPS)

  sets the max fps and the window to use
- ∼ScreenManager ()

  basic destructor
- Screen ∗ getCurrScreen () const

  returns the active screen
- void addScreen (Screen ∗newScreen, int escIndex=-1, int xIndex=-1, crstring title="")

  adds a screen and sets some of its properties
- void switchScreen (int newIndex)

  switches the to a new screen
- void destroy ()

  destroys screens
- void run ()

  runs screen logic (render, update, handleInput, etc.)

3.26.1 Detailed Description

Manages a collection of screens.

Definition at line 12 of file ScreenManager.h.

The documentation for this class was generated from the following files:

- include/nta/ScreenManager.h
- src/ScreenManager.cpp

3.27 nta::SoundEffect Class Reference

Represents a sound effect or short audio clip.

```
#include <AudioManager.h>
```

Public Member Functions

- void play (int numLoops=0) const

  plays the sound effect

Friends

- class **AudioManager**

3.27.1 Detailed Description

Represents a sound effect or short audio clip.

Definition at line 12 of file AudioManager.h.

The documentation for this class was generated from the following file:

- include/nta/AudioManager.h

3.28 nta::Sprite Class Reference

represents a textured quad

```
#include <Sprite.h>
```

Public Member Functions

- Sprite ()

  constructor and destructor
- void init (float x, float y, float w, float h, crstring imagePath, float d=0)

  creates the sprite
- void render () const

  renders the sprite

3.28.1 Detailed Description

represents a textured quad

Definition at line 10 of file Sprite.h.

3.28.2 Member Function Documentation

3.28.2.1 init()

```
void nta::Sprite::init (
            float x,
            float y,
            float w,
            float h,
            crstring imagePath,
            float d = 0 )
```

creates the sprite

first triangle

second triangle

Definition at line 11 of file Sprite.cpp.

The documentation for this class was generated from the following files:

- include/nta/Sprite.h
- src/Sprite.cpp

3.29 nta::SpriteBatch Class Reference

represents a collection of sprites to be drawn

`#include <SpriteBatch.h>`

Public Member Functions

- SpriteBatch ()

  constructor and destructor
- void init ()

  initializes the batch
- void begin ()

  begins collection of glyphs for the batch
- void end ()

  ends collection of glyphs and prepares to render
- void addGlyph (crvec4 posRect, crvec4 uvRect, GLuint texture, float depth=1, crvec4 color=glm::vec4(1))

  adds a glyph to the batch
- void **addGlyph** (crvec2 corner1, crvec2 corner2, crvec4 uvRect, GLuint texture, float depth=1, crvec4 color=glm::vec4(1))
- void render () const

  renders the batch

3.29.1 Detailed Description

represents a collection of sprites to be drawn

Definition at line 53 of file SpriteBatch.h.

The documentation for this class was generated from the following files:

- include/nta/SpriteBatch.h
- src/SpriteBatch.cpp

3.30 nta::SpriteFont Class Reference

Loads in a .ttf file, creates a font texture from it, and is then used to render text.

```
#include <SpriteFont.h>
```

Public Member Functions

- glm::vec2 measure (crstring text) const

  returns the dimensions of the rectangle containing the text
- void drawText (SpriteBatch &batch, crstring text, crvec2 topLeft, crvec2 scale, crvec4 color=glm::vec4(1), float depth=1) const

  renders text with specified location, color, scale, etc.
- void **drawText** (SpriteBatch &batch, crstring text, crvec4 posRect, crvec4 color=glm::vec4(1), float depth=1) const
- void drawTexture (SpriteBatch &batch) const

  renders texture

Public Attributes

- friend **ResourceManager**

3.30.1 Detailed Description

Loads in a .ttf file, creates a font texture from it, and is then used to render text.

Definition at line 52 of file SpriteFont.h.

The documentation for this class was generated from the following files:

- include/nta/SpriteFont.h
- src/SpriteFont.cpp

3.31 nta::SystemManager Class Reference

Static Public Member Functions

- static GLSLProgram $*$ **getGLSLProgram** (crstring progPath)
- static Window $*$ **getWindow** (crstring windowTitle, int flags=0)
- static void **destroy** ()

3.31.1 Detailed Description

Definition at line 10 of file SystemManager.h.

The documentation for this class was generated from the following files:

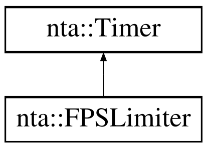
- include/nta/SystemManager.h
- src/SystemManager.cpp

3.32 nta::Timer Class Reference

represents a timer

```
#include <Timer.h>
```

Inheritance diagram for nta::Timer:



Public Member Functions

- Timer ()

  constructor and destructor
- virtual void begin ()

  begins timer
- virtual long double end () const

  return time since beginning of timer in nanoseconds

Protected Attributes

- std::chrono::time_point< std::chrono::high_resolution_clock, std::chrono::nanoseconds > **m_startPoint**

3.32.1 Detailed Description

represents a timer

Definition at line 8 of file Timer.h.

The documentation for this class was generated from the following files:

- include/nta/Timer.h
- src/Timer.cpp

3.33 nta::Vertex2D Struct Reference

represents a vertex in 2 dimensions

```
#include <Vertex.h>
```

Public Member Functions

- Vertex2D ()

  Initializes an "empty" vertex.
- Vertex2D (crvec2 p)

  Initializes a white, textureless vertex with given position.
- Vertex2D (crvec2 p, crvec4 c)

  Initialize textureless, colorful vertex.
- Vertex2D (crvec2 p, crvec4 c, crvec2 u, float t=1.0)

  Initialized a vertex with everything.
- void setPosition (float x, float y)

  sets the position of the vertex
- void setColor (float r, float g, float b, float a)

  sets the color of the vertex
- void **setColor** (crvec3 c)
- void setUV (float u, float v)

  sets the uv coordinates of the vertex

Public Attributes

- glm::vec2 pos

  the vertex's position, color, and uv coordinates, respectively
- glm::vec4 **color**
- glm::vec2 **uv**
- float **hasTexture**

3.33.1 Detailed Description

represents a vertex in 2 dimensions

Definition at line 11 of file Vertex.h.

The documentation for this struct was generated from the following file:

- include/nta/Vertex.h

3.34 nta::Window Class Reference

Represent a window.

```
#include <Window.h>
```

Public Member Functions

- Window ()
  constructor and destructor
- glm::vec2 getDimensions () const
  returns the window's dimensions
- std::string getTitle () const
  returns the window's title
- int getWidth () const
  returns the width of the window
- int getHeight () const
  returns the height of the window
- void setDimensions (int width, int height)
  updates the window's stored dimensions
- void swapBuffers () const
  updates the screen
- void screenshot () const
  stores a screenshot

Friends

- class **SystemManager**

3.34.1 Detailed Description

Represent a window.

Definition at line 13 of file Window.h.

The documentation for this class was generated from the following files:

- include/nta/Window.h
- src/Window.cpp

Index