AnimCreator – Technical Document

Duncan Bourquard

Date: 20.11.2018
Version: 1.0

Summary

Introduction

AnimCreator is a tool allowing the user to create sprite animations in the game enginge SFGE. It is developed to fill one of the summative of the module GPR 51100 of the courses Games Programming of the SAE Institute of Geneva. Its goal is to easily create animations from basic sprites or sprite sheets, to then export it to SFGE.

The tool is developed in C++, uses the libraries SFML and ImGui-SFML to deal with the graphic interface and Json-hpp to export to Json.

Specifications

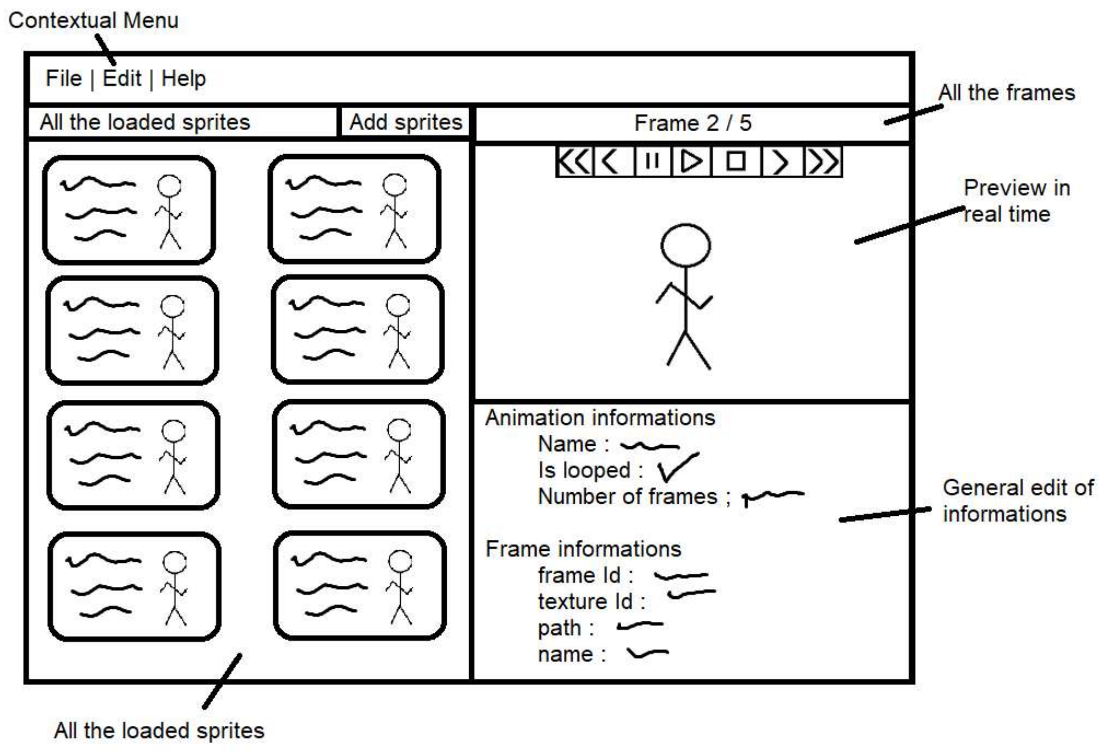
AnimCreator

The tool has the following functionalities:

(In italic are the optional functionalities)

- Import images inside the tool
    - Images can come from different folders.
    - Images can contain sprite sheets: Allow the user to define how many columns and rows they contain and the size of a single sprite in the sheet.
- Edit the animations frame perfect.
    - Define the speed of the animation
    - Define how many frames we want in the animation and assigns a texture to it.
    - Allowing to display the current state of the animation directly on screen.
- Export the animation under different formats
    - JSON, for SFGE
    - *GIF (for the user and possible other tools)*
- *Project*
    - *Save the project and edit it afterward, even after the application is closed.*
    - *Load a saved project in the application and gather all the information previously stored.*

UI structure



Many sections are defined, each fulfilling their tasks:

File Browser

Where all the loaded textures are displayed and usable. Their information, such as their name and location, are displayed next to each of them. When the user selects a texture, it will be highlighted and can directly be assigned to the current frame through a button or a double click. The texture can also be added at the end of the animation or deleted from the application.

Preview in real time

Where is displayed the current animation in real time. The user can navigate through the frames, exchange the order of them and add/deletes some.

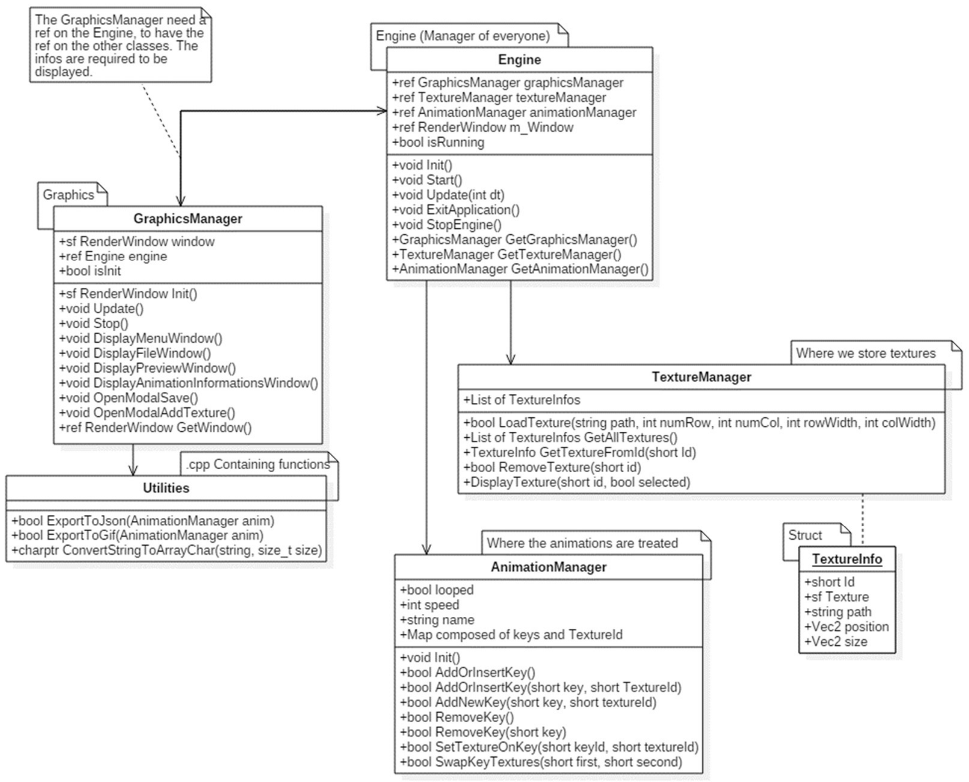
- << : Displays the first frame
- <+ : Add a frame before the currently displayed one
- < : Displays the frame before the currently displayed one
- II : Pauses the animations
- I> : Plays the animation
- ▪ : Stops the animation
- > : Display the frame after the currently displayed one
- +> : Add a frame after the currently displayed one
- >> : Displays the last frame

General information

Where all the information concerning the animation and the current displayed frame are noted in. The fields displayed are notably the name, if the animation is a loop or its speed in millisecond per frame.

Technical structure

Here is the UML diagram of the tool:


The GraphicsManager need a ref on the Engine, to have the ref on the other classes. The infos are required to be displayed.
Engine (Manager of everyone)
Engine
+ref GraphicsManager graphicsManager
+ref TextureManager textureManager
+ref AnimationManager animationManager
+ref RenderWindow m_Window
+bool isRunning
+void Init()
+void Start()
+void Update(int dt)
+void ExitApplication()
+void StopEngine()
+GraphicsManager GetGraphicsManager()
+TextureManager GetTextureManager()
+AnimationManager GetAnimationManager()
Graphics
GraphicsManager
+sf RenderWindow window
+ref Engine engine
+bool isInit
+sf RenderWindow Init()
+void Update()
+void Stop()
+void DisplayMenuWindow()
+void DisplayFileWindow()
+void DisplayPreviewWindow()
+void DisplayAnimationInformationsWindow()
+void OpenModalSave()
+void OpenModalAddTexture()
+ref RenderWindow GetWindow()
.cpp Containing functions
Utilities
+bool ExportToJson(AnimationManager anim)
+bool ExportToGif(AnimationManager anim)
+charptr ConvertStringToArrayChar(string, size_t size)
Where we store textures
TextureManager
+List of TextureInfos
+bool LoadTexture(string path, int numRow, int numCol, int rowWidth, int colWidth)
+List of TextureInfos GetAllTextures()
+TextureInfo GetTextureFromId(short Id)
+bool RemoveTexture(short id)
+DisplayTexture(short id, bool selected)
Where the animations are treated
AnimationManager
+bool looped
+int speed
+string name
+Map composed of keys and TextureId
+void Init()
+bool AddOrInsertKey()
+bool AddOrInsertKey(short key, short TextureId)
+bool AddNewKey(short key, short textureId)
+bool RemoveKey()
+bool RemoveKey(short key)
+bool SetTextureOnKey(short keyId, short textureId)
+bool SwapKeyTextures(short first, short second)
Struct
TextureInfo
+short Id
+sf Texture
+string path
+Vec2 position
+Vec2 size

Engine

The Engine manages every classes needed to make the tool work properly. It initializes every of them and manage the global update of the engine. The engine needs to be initialized before starting.

GraphicsManager

The GraphicsManager manages and display all the information on screen. It contains all the UI construction, and deals with the user's action.

TextureManager

Where every texture is loaded and stored. The other classes can get the full list or any of them by knowing its Id.

TextureInfo

Structure of the texture's information.

AnimationManager

Where the animation is managed and stored. It deals with texture assignment to a keyframe and contains the general information of the animation.

Utilities

A class used to bring utility functions to the application, such as the exportation or the conversion of datatype into a different form.

Structure of the JSON file

```
{
        ″name″: string,
        ″speed″: unsigned int,
        ″isLooped″:                                     Boolean,
        ″frames″ : [
                {
                        ″key″: int,
                        ″filename″: string,
                        ″Position of the texture″: {x, y},
                        ″Size of the texture″: {x, y}
                },
                {
                        Other frames…
                }
        ]
}
```

SFGE side

A new Component, named Animation2d, have been added to the game engine. It works the same way as a the Sprite2d, but with switches of textures.

A test has been implemented to try the animation on the engine, named “test_animation”.

Vocabulary

• Animation:	Result wanted from the tool. It contains Keyframes and other informations.
• Keyframe:	A key in the animation, representing a frame.
• Sprite / Texture:	Image that can be assigned to a keyframes.

Conclusion

The tool functions well and is delivered. I’m personally proud of what I could achieve in the little time I had, and I plan on improving it and creating other useful tools in the future.

How to go further

The UI side lacks a bit of functionality. Little functionalities can easily be added, and the buttons can be better organized to clarify the situation. An exportation to Gif and the opening of recent projects are the main features the tool lacks.