ASSIGNMENT 2 - ALPHA COMPOSITING

5SD814: Programming in C/C++ 4: Computer Games in 2D - 2020

Introduction

Software blitting is a technique of combining multiple bitmap images using either boolean functions or alpha compositing. This assignment will focus on the latter.

Understanding the fundamentals of software blitting and how it can be applied is still highly relevant and important. Even in this day and age of highly specialized and parallel hardware like the Graphics Processing Units (GPU) - a piece of hardware that nearly every kitchen appliance and garden rake seem to have nowadays. GPUs have massive feature sets and immense speeds - speeds which are in the order of magnitudes higher - compared to doing the equivalent using the Central Processing Unit (CPU).

The technique of software blitting is still relevant in cases where a GPU is not present, as a fallback for debugging certain visual features, when implementing complicated formulas or when there is a need for higher precision of the pixel data for visual analytics e.g. x-ray imagery in the medical industry.

Description

This is an *individual* assignment where you implement low level software blitting (sometimes called software rendering) using alpha compositing with raw image data from a spritesheet tileset image loaded from disk.

An overview of features to implement:

1. Create a Bitmap with a user specified width and height
2. Determine if a pixel position is inside a Bitmap
3. Get the Color at a specific pixel position
4. Set the Color of a specific pixel at a specific position
5. Clear all pixels to a user specified Color
6. Draw a portion of a source Bitmap into a destination Bitmap at a position
7. Draw the explosion animation looping

Features

All features below should be implemented in `app.hpp` which contains all function shims - yes, in the header file! All function shims below should be defined in the correct order as defined in the header file.

1. Create a Bitmap

Allocate the appropriate amount of dynamic memory for the Bitmap data (essentially `new Color[...]`). User provided width and height must be used.
Must be a 1-dimensional array.

2. Determine Pixel Containment

Determine if the user provided position x and y are inside the containment of the Bitmap (i.e width and height).

3. Get Pixel Color

Return the pixel color from the user specified position. Return magenta if the Bitmap is invalid or the position is not inside the Bitmap.

4. Set Pixel Color

Set the pixel color to the user specified Color at the requested position.
If the Bitmap is invalid or the position is not inside the Bitmap - do nothing.

5. Clear Bitmap

Set all valid pixels in the Bitmap to the user specified Color.

6. Blit Bitmap Region

This is the core function of the assignment: Blit a user specified portion of a source bitmap onto a destination bitmap at a user specified position.

Blitting must take the alpha values of the source image into consideration when applying a pixel color value to the destination, i.e. if alpha for a pixel is zero the pixel components are not applied to the destination. Also, make sure to verify that the user provided source rectangle is valid, i.e. is not bigger than the source image or that the rectangles width and height is not negative etc.

7. Draw Animation Looping

The loaded bitmap image contains 12 frames of an explosion, each frame is 96x96 pixels. Draw the animation looping somewhere on screen.

Each animation frame should be displayed for 100ms (the application updates roughly 60 times per second, which gives us ~17ms per frame). Clear the bitmap each application update.

Requirements

Original code written by the student using C++. The assignment must use the `lime` project. All features must be implemented for a *Pass*. Cannot manipulate or change a single pixel in the image file. Use warning level 4 and treat warnings as errors where only the following warnings can be disabled: 4100, 4189 and 4505. Window resolution or image dimensions cannot be changed.

Lecture material, external content and various coding snippets and solutions provided by the course responsible and teachers can and should be used when implementing the assignment.

Grading

The grades that can be achieved in this assignment are *Pass* and *Fail*. Successfully completing all listed features and requirements will earn you a *Pass*.

Hand in

Upload the project to the course page on Studentportalen in a ***zip***-file. Name the file as follows: `5sd814-assignment-3-surname-firstname.zip` (it should be fairly obvious that you should replace surname and firstname with your own dito).

The hand in must also include a *readme.txt* that should contain the course name (with course code), the name of the student and a short assignment description and

a short usage description if applicable. Place this description text file in the root folder of the project.

The text file should also contain the name/names of other students if you have collaborated with; i.e. you are allowed - and encouraged - to talk to other students about the implementations of the assignment. But obviously you should implement it yourself.

Please remember to remove all redundant build files and folders from the project before uploading. This includes all folders generated by Visual Studio when opening and compiling one or more projects, i.e. the hidden `.vs` and `.git` folders and the `build` folder.

Failing to comply with ***any*** of the assignment or hand in requirements will result in a *Fail* on the assignment.