06 Texts

Functional and Implementation Guidelines

Functional Guidelines

1. Your application should support dynamic loading of fonts
   1. The same font should be able to be loaded with different font size
2. Your application should implement dynamic texts. Those texts should support:
   1. the ASCII codes that can be represented
   2. hardware acceleration (GPU acceleration)
   3. different colors
   4. scaling and alpha blending
   5. anti-aliasing (no sharp edges)
3. Try to incorporate the Flyweight design pattern for the texts as well
   1. This would mean that the “text object” should not carry inside the massive graphical information inside of it
   2. There should be a “central” run-time repository for texts storing/loading

Implementation Guidelines

1. **True Type Fonts (TTF) - Lecture**
2. TTF usage
   1. Experiment - Implement a “loadText” solution directly in the Engine.cpp file.
   2. OpenFont

| TTF\_Font \* font = TTF\_OpenFont(“path-to-font”, (int)fontSize); |
| --- |

* 1. SDL\_Color

| SDL\_Color color = {.r = 127, .g = 127, .b = 127, .a = 255}; |
| --- |

* 1. TTF\_RenderText\_Solid

| SDL\_Surface \*textSurface = TTF\_RenderText\_Solid(font, text, &color); |
| --- |

* 1. save width, heigh

| const int32\_t textWidth = textSurface>width;  const int32\_t textHeight = textSurface>height; |
| --- |

* 1. Create TextureFromSurface

| SDL\_Texture\* texture = SDL\_CreateTextureFromSurface(renderer, textSurface); |
| --- |

* 1. Free surface and close font

| SDL\_FreeSurface(textSurface); TTF\_CloseFont(font); |
| --- |

1. TTF\_RenderText\_Solid

| SDL\_Surface \*textSurface = TTF\_RenderText\_Blended(font, text, &color); |
| --- |

1. Expand the Texture set of functions - loadSurfaceFromText()
2. Implement Color class

| struct RGBA {  uint8\_t r { 0 };  uint8\_t g { 0 };  uint8\_t b { 0 };  uint8\_t a { 255 };  };  class Color {  Color(const uint8\_t red, const uint8\_t green, const uint8\_t blue,  const uint8\_t alpha = 255);  //forbid default constructor  Color() = delete;  RGBA rgba;  }; |
| --- |

* 1. Export the most common colors

| namespace Colors {  const Color RED(255, 0, 0, 255);  const Color GREEN(0, 255, 0, 255);  const Color BLUE(0, 0, 255, 255);  const Color BLACK(0, 0, 0, 255);  const Color WHITE(255, 255, 255, 255);  const Color GRAY(192, 192, 192, 255);  const Color CYAN(0, 255, 255, 255);  const Color MAGENTA(255, 0, 255, 255);  const Color YELLOW(255, 255, 0, 255);  const Color ORANGE(255, 128, 0, 255);  const Color PURPLE(128, 0, 128, 255);  const Color FULL\_TRANSPARENT(0, 0, 0, 0);  } //namespace Colors |
| --- |

1. Expand the main application window to 800/600
2. Expand the DrawParams struct by adding enum WidgetType (IMAGE or TEXT)

| enum class WidgetType : uint8\_t {  IMAGE,  TEXT,  UNKNOWN  }; |
| --- |

1. Expand the Renderer to have public API method drawWidget

| void drawWidget(const DrawParams& drawParams, SDL\_Texture \*texture); |
| --- |

* 1. create private methods

| void drawImage(const DrawParams& drawParams, SDL\_Texture \*texture);  void drawText(const DrawParams& drawParams, SDL\_Texture \*texture); |
| --- |

1. Implement TextContainer
   1. Structure

| class TextContainer {  //the textures we'll be drawing  std::vector<SDL\_Texture\*> \_textures;  std::unordered\_map<int32\_t, TTF\_Font\*> \_fonts;  }; |
| --- |

* 1. TextContainerCfg - use the analogy from ImageContainerCfg to create one for fonts
  2. Unlike images - texts can not be easily reused. Each text is unique.
  3. Public API

| void createText(const char \*text, const Color &color, int32\_t fontId,  int32\_t &outTextId, int32\_t &outTextWidth,  int32\_t &outTextHeight);  void reloadText(const char \*text, const Color &color, int32\_t fontId,  int32\_t textId, int32\_t &outTextWidth,  int32\_t &outTextHeight);  void unloadText(int32\_t textId);  SDL\_Texture\* getTextTexture(int32\_t textId) const; |
| --- |

* 1. Implement Helper methods - occupyFreeSlotIndex(), freeSlotIndex()
  2. BONUS: For homework make increase the lookup speed for occupyFreeSlotIndex() from O(N) Linear to O(1) constant time

1. Add several Texts in the Game class (as DrawParams)
   1. For example, pressing the M key will hide one text and draw another
   2. Pressing the N key will show the first text and hide the other