

VroemVroem - Class diagram

Game
<div>- <u>instance</u>: Game *</div> <div>- title: const char *</div> <div>- width: int</div> <div>- height int</div> <div>- fullscreen: bool</div> <div>- running: bool = false</div> <div>- time: uint64_t = 0</div> <div>- window: std::unique_ptr<SDL_Window, SDL_Deleter></div> <div>- canvas: std::shared_ptr<Canvas></div> <div>- page: std::unique_ptr<Pages::Page></div>
<div>+ Game(const char *, int, int, bool)</div> <div>+ <u>getInstance(): Game *</u></div> <div>+ getTitle(): const char *</div> <div>+ getWidth(): int</div> <div>+ getHeight(): int</div> <div>+ getFullscreen(): bool</div> <div>+ setFullscreen(bool)</div> <div>+ getTime(): uint64_t</div> <div>+ getCanvas(): std::shared_ptr<Canvas></div> <div>+ getPage(): Pages::Page *</div> <div>+ setPage(std::unique_ptr<Pages::Page>)</div> <div>+ start()</div> <div>+ stop()</div> <div>- handleEvent(const SDL_Event *): bool</div>

Config
<div>+ <u>name</u>: const char *</div> <div>+ <u>version</u>: const char *</div> <div>+ <u>gitRepoUrl</u>: const char *</div>

Rect
<div>+ x: int</div> <div>+ y: int</div> <div>+ width: int</div> <div>+ height: int</div>
<div>+ Rect()</div> <div>+ Rect(int, int, int, int)</div> <div>+ containsPoint(int, int): bool</div> <div>+ collides(Rect *): bool</div>

Color
<div>+ red: uint8_t</div> <div>+ blue: uint8_t</div> <div>+ green: uint8_t</div> <div>+ alpha: uint8_t</div>
<div>+ Color()</div> <div>+ Color(uint8_t, uint8_t, uint8_t)</div> <div>+ Color(uint8_t, uint8_t, uint8_t, uint8_t)</div>

Camera
<div>+ <u>zoomLevels</u>: int[]</div> <div>+ <u>zoomLevelsSize</u>: int</div> <div>- x: float</div> <div>- y: float</div> <div>- width: int</div> <div>- height: int</div> <div>- zoom: int</div> <div>- drag: struct<div>- enabled: bool</div><div>- begin: struct<div>- x: float</div><div>- y: float</div></div><div>- mouse: struct<div>- x: int</div><div>- y: int</div></div></div>
<div>+ Camera(const WorldInfo *, float, float, int, int, int)</div> <div>+ getX(): float</div> <div>+ setX(float)</div> <div>+ getY(): float</div> <div>+ setY(float)</div> <div>+ getWidth(): int</div> <div>+ getHeight(): int</div> <div>+ getZoom(): int</div> <div>+ setZoom(int)</div> <div>+ handleEvent(const SDL_Event *): bool</div>

World
<div>- seed uint64_t</div> <div>- width: int</div> <div>- height: int</div> <div>- random: std::unique_ptr<Random></div> <div>- terrainMap: std::unique_ptr<uint8_t[]></div> <div>- objectMap: std::unique_ptr<uint8_t[]></div> <div>- natures: std::vector<std::unique_ptr<Nature>></div> <div>- houses: std::vector<std::unique_ptr<House>></div> <div>- cities: std::vector<std::unique_ptr<City>></div> <div>- roads: std::vector<std::unique_ptr<Road>></div> <div>- vehicles: std::vector<std::unique_ptr<Vehicle>></div>
<div>+ World(uint64_t, int, int)</div> <div>+ getSeed(): uint64_t</div> <div>+ getWidth(): int</div> <div>+ getHeight(): int</div> <div>+ getNatures(): std::vector<const Nature *></div> <div>+ getHouses(): std::vector<const House *></div> <div>+ getCities(): std::vector<const City *></div> <div>+ getRoads(): std::vector<const Road *></div> <div>+ getVehicles(): std::vector<const Vehicle *></div> <div>+ update(float)</div> <div>+ draw(std::shared_ptr<Canvas>, const Camera *)</div>

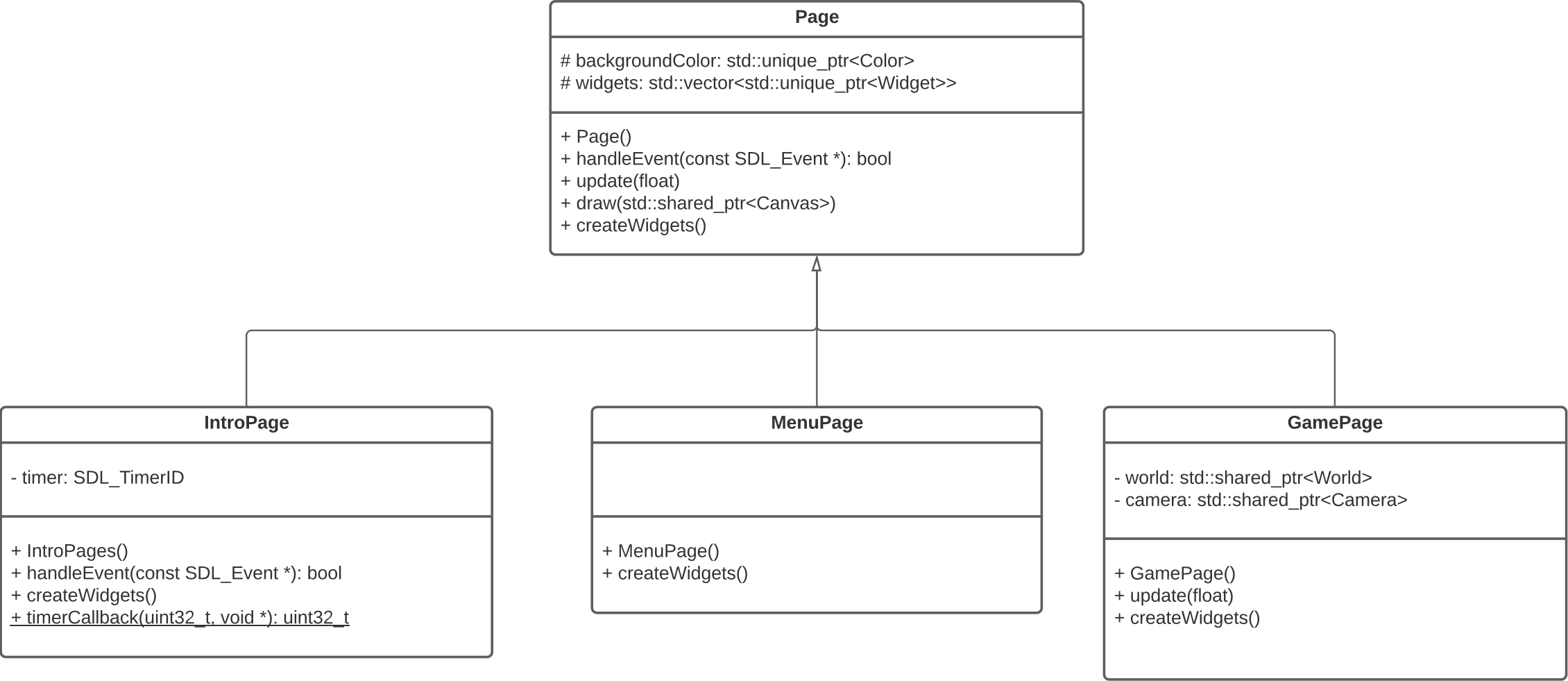
Canvas
<div>- renderer: std::unique_ptr<SDL_Renderer></div>
<div>+ Canvas(SDL_Window *)</div> <div>+ getRenderer(): SDL_Renderer *</div> <div>+ getRect(): std::unique_ptr<Rect></div> <div>+ clear(const Color *)</div> <div>+ fillRect(const Rect *, const Color *)</div> <div>+ drawTexture(SDL_Texture *, const Rect *)</div> <div>+ drawTexture(SDL_Texture *, const Rect *, float)</div> <div>+ present()</div>

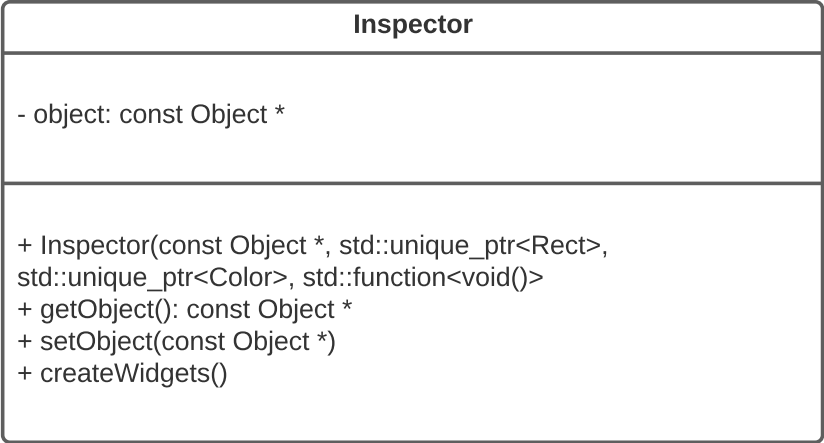
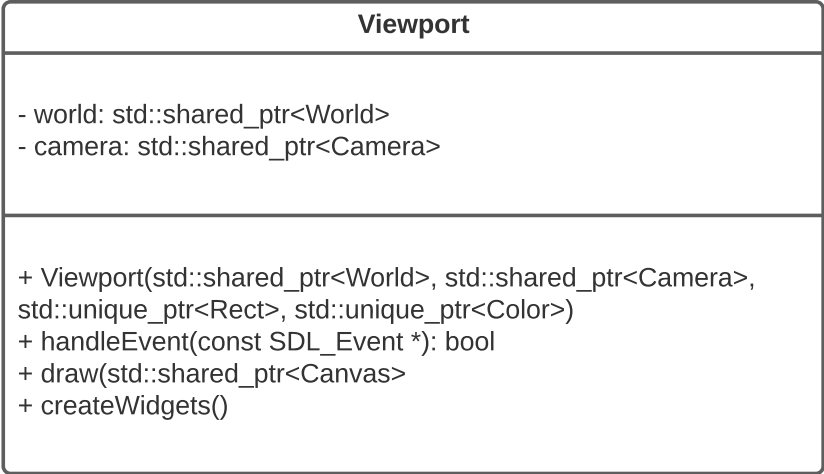
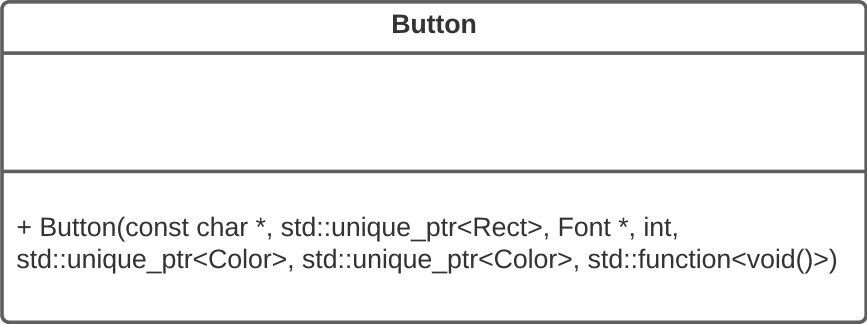
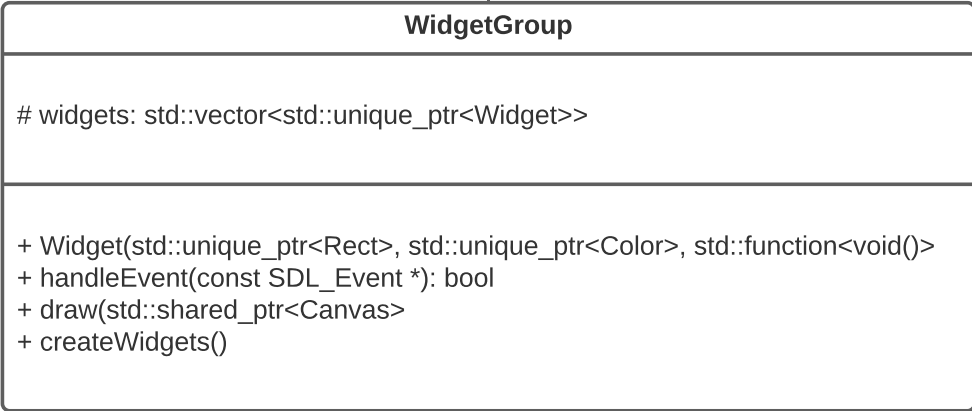
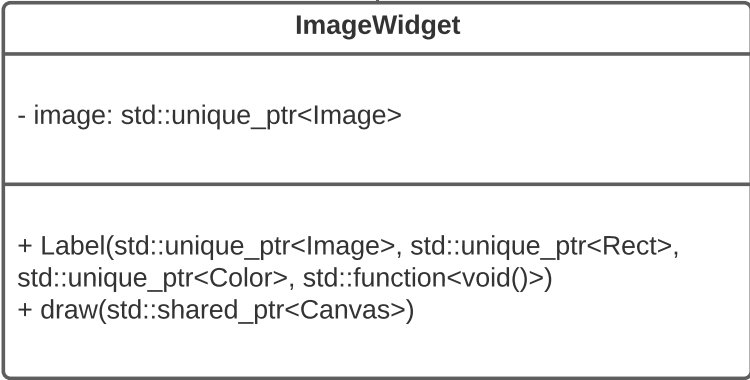
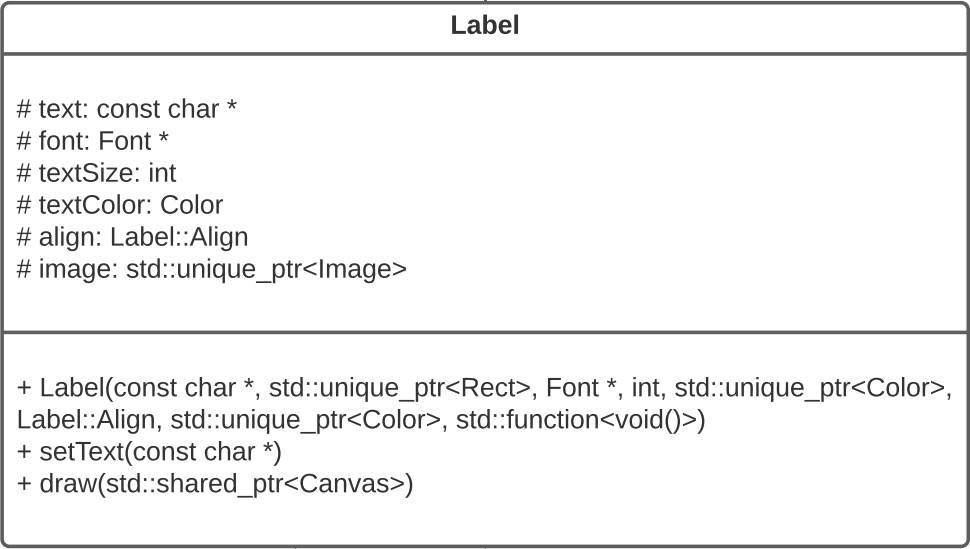
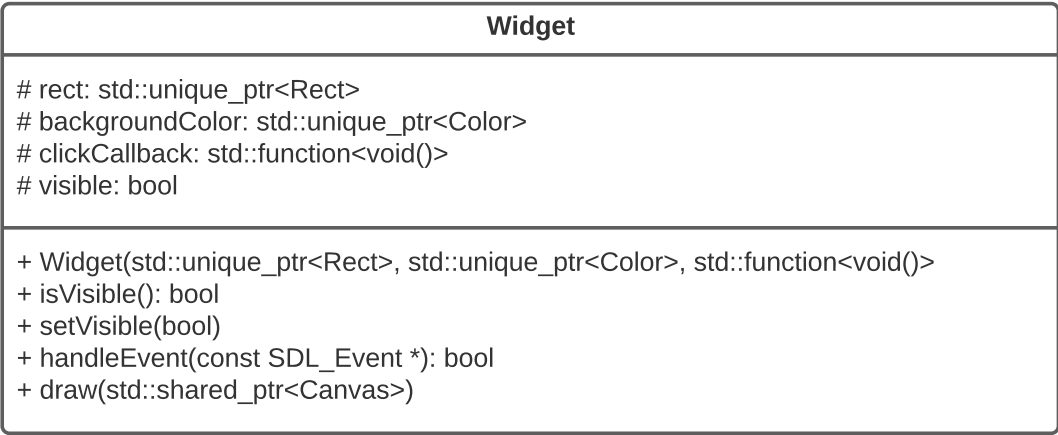
Fonts
<div>- <u>instance</u>: unique_ptr<Fonts></div> <div>- titleFont: std::unique_ptr</div> <div>- textFont: std::unique_ptr</div>
<div>+ Fonts()</div> <div>+ <u>getInstance(): Fonts *</u></div> <div>+ getTitleFont(): Font *</div> <div>+ getTextFont(): Font *</div>

Font
<div>- fontBuffer: std::unique_ptr<uint8_t[]></div> <div>- fontInfo: stbtt_fontinfo</div>
<div>+ Font(const char *)</div> <div>+ measure(const char *, int): int</div> <div>+ render(std::shared_ptr<Canvas>, const char *, int, uint32_t): std::unique_ptr<Image></div>

Image
<div>- canvas: std::shared_ptr<Canvas></div> <div>- width: int</div> <div>- height: int</div> <div>- transparent: bool</div> <div>- texture: std::unique_ptr<SDL_Texture, SDL_Deleter></div>
<div>+ Image(std::shared_ptr<Canvas>, const char *, bool)</div> <div>+ Image(std::shared_ptr<Canvas>, int, int, bool, const uint8_t *)</div> <div>+ getCanvas(): std::shared_ptr<Canvas></div> <div>+ getWidth(): int</div> <div>+ getHeight(): int</div> <div>+ isTransparent(): bool</div> <div>+ getTexture(): SDL_Texture *</div> <div>+ draw(const Rect *)</div> <div>+ draw(const Rect *, float)</div> <div>- loadBitmap(const uint8_t *)</div>

Random
<div>- <u>instance</u>: std::unique_ptr<Random></div> <div>- seed: uint64_t</div>
<div>+ Random(uint64_t)</div> <div>+ <u>getInstance(): Random *</u></div> <div>+ setSeed(uint64_t)</div> <div>+ getSeed(): uint64_t</div> <div>+ random(): double</div> <div>+ random(int, int): int</div>





Objects

