Funkcje oznaczone kolorem magenta są wirtualne, a funkcje oznaczone kolorem fioletowym są abstrakcyjne. Z kolei funkcje i atrybuty oznaczone kolorem pomarańczowym zostaną zmienione. Oznaczenia kolorem niebieskim oznaczają statyczność.

(!) oznacza, że wartość zwrócona nie powinna być zignorowana. "u8", "u16", "u32", "u64", "i8", "i16", "i32" oraz "i64" to skróty od typów liczbowych standardu POSIX.

ResourceManager

- textures : vector<SDL_Texture*>
- soundEffects : vector<Mix_Chunk*>
- music : vector<Mix_Music*>
- latestStatus : Status

+ ~ResourceManager() : void
+ init() : Status
+ getLatestStatus() : Status
+ registerTexture(path) : u32
+ getTextureHandle(id: u32) : TextureHandle
+ getTexture(id: u32) : SDL_Texture*
+ getTextureOriginalSize(id: u32) : Size
- createFallbackTexture() : SDL_Texture*

Logger

- stream : std::ofstream
- latest : time_t
- date : char[20]

+ ~Logger() : void
- setDateString(full?) : void
- checkTimestamp() : bool
+ printShared() : void
+ getDateString(full?) : C-String
+ getCurrentDateString(full?) : C-String
+ init(pathToFile : String) : Status (!)
+ init(pathToFile : C-String) : Status (!)
+ flush() : void
+ print(Args...) : void
+ debug(Args...) : void
+ info(Args...) : void
+ warn(Args...) : void
+ error(Args...) : void
+ fatal(Args...) : void
+ trace(Args...) : void

InputHandler

- latestEvents : EventBuffer

+ getLatestEvent() : const SDL_Event&
+ processInput(game: Game*) : void

GameObject

uuid : UUID
flags : u64
name : C-String

+ GameObject(objectID: u32, name: C-String) : this
+ GameObject(objectID: u32) : this
+ ~GameObject() : void
+ getName() : C-String
# setName(name: String) : void
# setName(name: C-String) : void
+ getUUID() : u64
# setUUID(uuid: u64) : void
+ getInstanceID() : u32
# setInstanceID(instance: u32) : void
+ getTypeID() : u32
# setTypeID(type: u32) : void
+ getFlags() : u64
# setFlag(flag: u32) : void
# setFlags(flags: u64) : void
# clearFlag(flag: u32) : void
# clearFlags() : void
# flipFlag(flag: u32) : void
+ isFlagSet(flag: u32) : bool

resourceManager

logger

inputHandler

Program

+ keyboardState: u8
# flags : ProgramFlags
# numberOfKeyboardKeys : u32
# clockFrequency: u64
# timeSinceStart : u64
# window : SDL_Window*
# windowParameters: WindowParameters
# renderingContext: SDL_Renderer*
# audioParameters : AudioParameters
# logger: Logger
# resourceManager: ResourceManager

+ *sim*Program() : void
+ initSystems() : Status (!)
+ getClockFrequency() : u64
+ getRenderingContext() : SDL_Renderer*
+ getLogger() : Logger&
+ getResourceManager() : ResourceManager&
+ getScreenSize() : Size
+ getNumberOfKeys() : u32

RenderableObjectBase

- textureHandle : TextureHandle

+ RenderableObjectBase(objectID: u32) : this
+ RenderableObjectBase(objectID: u32, name: C-String) : this
+ RenderableObjectBase(objectID: u32, textureHandleIndex: u32) : this
+ RenderableObjectBase(objectID: u32, name: C-String, textureHandleIndex: u32) : this
+ render() : void
+ isVisible() : bool
+ setVisible() : void
+ setInvisible() : void
+ flipVisible() : void
+ getTexture() : SDL_Texture*
+ bindTexture(textureHandleIndex: u32) : this

World

chunks : HashMap<Point, Chunk*>

+ World(objectID: u32) : this
+ World(objectID: u32, name: C-String) : this
+ ~World() : void
+ getChunk(which: Point) : const Chunk*
+ getChunk(x: i32, y: i32) : const Chunk*
+ populateChunk(which: Point, blockID: u32) : Status
+ getBlockAt(x: i32, y: i32) : const Block*
+ getBlockAt(pos: BlockPos) : const Block*
+ printChunk(x: i32, y: i32) : void

Block

blockAABB: AABB

+ Block(objectID: u32) : this
+ Block(objectID: u32, name: C-String) : this
+ Block(objectID: u32, textureHandleIndex: u32) : this
+ Block(objectID: u32, name: C-String, textureHandleIndex: u32) : this
+ render() : void (override)

RenderableObject

texturePortion : rectangle
targetPortion : rectangle
angle : double
flip : SDL_RendererFlip

+ RenderableObject(objectID: u32) : this
+ RenderableObject(objectID: u32, name: C-String) : this
+ RenderableObject(objectID: u32, textureHandleIndex: u32) : this
+ RenderableObject(objectID: u32, name: C-String, textureHandleIndex: u32) : this
+ getTexturePortion() : const rectangle&
+ setTexturePortion(r: rectangle) : this
+ setTexturePortionOriginal() : void
+ getTargetPortion() : const rectangle&
+ setTargetPortion(r: rectangle) : this
+ scaleX(scale: float) : this
+ scaleY(scale: float) : this
+ scale(scale: float) : this
+ scaleX(scaleX: float, scaleY: float) : this
+ rotate(degrees: double) : void
+ setPositionOnScreen(x: int, y: int) : this
+ setPositionOnScreenCentered(x: int, y: int) : this
+ setSizeOnScreen(width: u32, height: u32) : this
+ unflip() : this
+ flipHorizontally() : this
+ flipVertically() : this

Game

- registry: MainRegistry
- renderer: GameRenderer
- inputHandler: InputHandler
- world : World

+ getRegistry() : MainRegistry&
+ getRenderer() : GameRenderer&
+ getInputHandler() : InputHandler&
+ init() : Status (!)
+ run() : void
+ getTimeSinceLastFrame() : u64
+ getWorld() : World&

UIElement

+ UIElement(objectID: u32) : this
+ UIElement(objectID: u32, name: C-String) : this
+ UIElement(objectID: u32, textureHandleIndex: u32) : this
+ UIElement(objectID: u32, name: C-String, textureHandleIndex: u32) : this
+ onHover() : void
+ onKeyPress(key: SD_Keycode) : void
+ onMouseClick(button: u8) : void

renderer

registry

GameRenderer

- uiElements : ListArray<UIElement*>
- fps : u32
- start: i64
- end: i64
- delta: i64
- overhead: i64
- frameTime: i64
- scalingFactor : double
- lastFrameAt: u64
- numberOfFramesRendered : u64
- cameraPosition : Point

+ GameRenderer() : this
- moveCamera(offX: i32, offY: i32) : void
+ registerUIElement(element: UIElement*) : void
+ renderInPlace(game: Game*) : void
+ getTimeSinceLastFrame() : u64
+ getCameraPosition() : Point

MainRegistry

- overrides : BitArray

+ MainRegistry() : this
+ ~MainRegistry() : void
+ init() : void

calls Blocks::init

gets blockID for init

Blocks

- blocks: vector<Block*>
- numberOfBlocksRegistered: u32

+ addBlock(block: Block*) : void
+ getCurrentBlockID() : u32
+ getBlockWithID(id: u32) : Block*
+ init(blockID: u32) : u32