VroemVroem v0.1.0 - Class diagram

Game

- instance: Game *
- title: const char *
- width: int
- height int
- fullscreen: bool
- running: bool = false
- time: uint64_t = 0
- window: std::unique_ptr<SDL_Window, SDL_Deleter>
- canvas: std::shared_ptr<Canvas>
- page: std::unique_ptr<Pages::Page>

+ Game(const char *, int, int, bool)
+ getInstance(): Game *
+ getTitle(): const char *
+ getWidth(): int
+ getHeight(): int
+ getFullscreen(): bool
+ setFullscreen(bool)
+ getTime(): uint64_t
+ getCanvas(): std::shared_ptr<Canvas>
+ getPage(): Pages::Page *
+ setPage(std::unique_ptr<Pages::Page>)
+ start()
+ stop()
- handleEvent(const SDL_Event *): bool

Config

+ name: const char *
+ version: const char *
+ gitRepoUrl: const char *
+ vehicleTimeout: int

Rect

+ x: int
+ y: int
+ width: int
+ height: int

+ Rect()
+ Rect(int, int, int, int)
+ containsPoint(int, int): bool
+ collides(Rect *): bool
+ collides(Rect *, float): bool

Timer

+ callback(uint32_t, void *): uint32_t

Camera

+ zoomLevels: int[]
+ zoomLevelsSize: int
- x: float
- y: float
- width: int
- height: int
- zoom: int
- mouseX: int
- mouseY: int
- drag: struct
  - enabled: bool
  - begin: struct
    - x: float
    - y: float
  - mouse: struct
    - x: int
    - y: int

+ Camera(const WorldInfo *, float, float, int, int, int)
+ getX(): float
+ setX(float)
+ getY(): float
+ setY(float)
+ getWidth(): int
+ getHeight(): int
+ getZoom(): int
+ setZoom(int)
+ getMouseX(): int
+ getMouseY(): int
+ handleEvent(const SDL_Event *): bool

World

- vehicleTimerEventCode: int
- seed uint64_t
- width: int
- height: int
- random: std::unique_ptr<Random>
- terrainMap: std::unique_ptr<uint8_t[]>
- objectMap: std::unique_ptr<uint8_t[]>
- natures: std::vector<std::unique_ptr<Nature>>
- houses: std::vector<std::unique_ptr<House>>
- cities: std::vector<std::unique_ptr<City>>
- roads: std::vector<std::unique_ptr<Road>>
- vehicles: std::vector<std::unique_ptr<Vehicle>>
- explosions: std::vector<std::unique_ptr<Explosion>>

+ World(uint64_t, int, int)
+ getSeed(): uint64_t
+ getWidth(): int
+ getHeight(): int
+ getNatures(): std::vector<const Nature *>
+ getHouses(): std::vector<const House *>
+ getCities(): std::vector<const City *>
+ getRoads(): std::vector<const Road *>
+ getVehicles(): std::vector<const Vehicle *>
+ getExplosions(): std::vector<const Explosion *>
+ handleEvent(const SDL_Event *): bool
+ update(float)
+ draw(std::shared_ptr<Canvas>, const Camera *)

Canvas

- renderer: std::unique_ptr<SDL_Renderer>

+ Canvas(SDL_Window *)
+ getRenderer(): SDL_Renderer *
+ getRect(): std::unique_ptr<Rect>
+ clear(const Color *)
+ fillRect(const Rect *, const Color *)
+ drawTexture(SDL_Texture *, const Rect *)
+ drawTexture(SDL_Texture *, const Rect *, float)
+ drawTexture(SDL_Texture *, const Rect *, const Rect *)
+ drawTexture(SDL_Texture *, const Rect *, const Rect *, float)
+ drawLine(int x0, int y0, int x1, int y1, const Color *)
+ present()

Fonts

- instance: unique_ptr<Fonts>
- titleFont: std::unique_ptr<Font>
- textFont: std::unique_ptr<Font>

+ Fonts()
+ getInstance(): Fonts *
+ getTitleFont(): Font *
+ getTextFont(): Font *

Color

+ red: uint8_t
+ blue: uint8_t
+ green: uint8_t
+ alpha: uint8_t

+ Color()
+ Color(uint8_t, uint8_t, uint8_t)
+ Color(uint8_t, uint8_t, uint8_t, uint8_t)

Image

- canvas: std::shared_ptr<Canvas>
- width: int
- height: int
- transparent: bool
- texture: std::unique_ptr<SDL_Texture, SDL_Deleter>

+ Image(std::shared_ptr<Canvas>, const char *, bool)
+ Image(std::shared_ptr<Canvas>, int, int, bool, const uint8_t *)
+ getCanvas(): std::shared_ptr<Canvas>
+ getWidth(): int
+ getHeight(): int
+ isTransparent(): bool
+ getTexture(): SDL_Texture *
+ draw(const Rect *)
+ draw(const Rect *, float)
+ draw(const Rect *, const Rect *)
+ draw(const Rect *, const Rect *, float)
- loadBitmap(const uint8_t *)

Font

- fontBuffer: std::unique_ptr<uint8_t[]>
- fontInfo: stbtt_fontinfo

+ Font(const char *)
+ measure(const char *, int): int
+ render(std::shared_ptr<Canvas>, const char *, int, uint32_t): std::unique_ptr<Image>

Random

- instance: std::unique_ptr<Random>
- seed: uint64_t

+ Random(uint64_t)
+ getInstance(): Random *
+ setSeed(uint64_t)
+ getSeed(): uint64_t
+ random(): double
+ random(int, int): int

Pages

Page

backgroundColor: std::unique_ptr<Color>
widgets: std::vector<std::unique_ptr<Widget>>

+ Page()
+ handleEvent(const SDL_Event *): bool
+ update(float)
+ draw(std::shared_ptr<Canvas>)
+ createWidgets()

IntroPage

- timerEventCode: int
- timer: SDL_TimerID

+ IntroPages()
+ handleEvent(const SDL_Event *): bool
+ createWidgets()

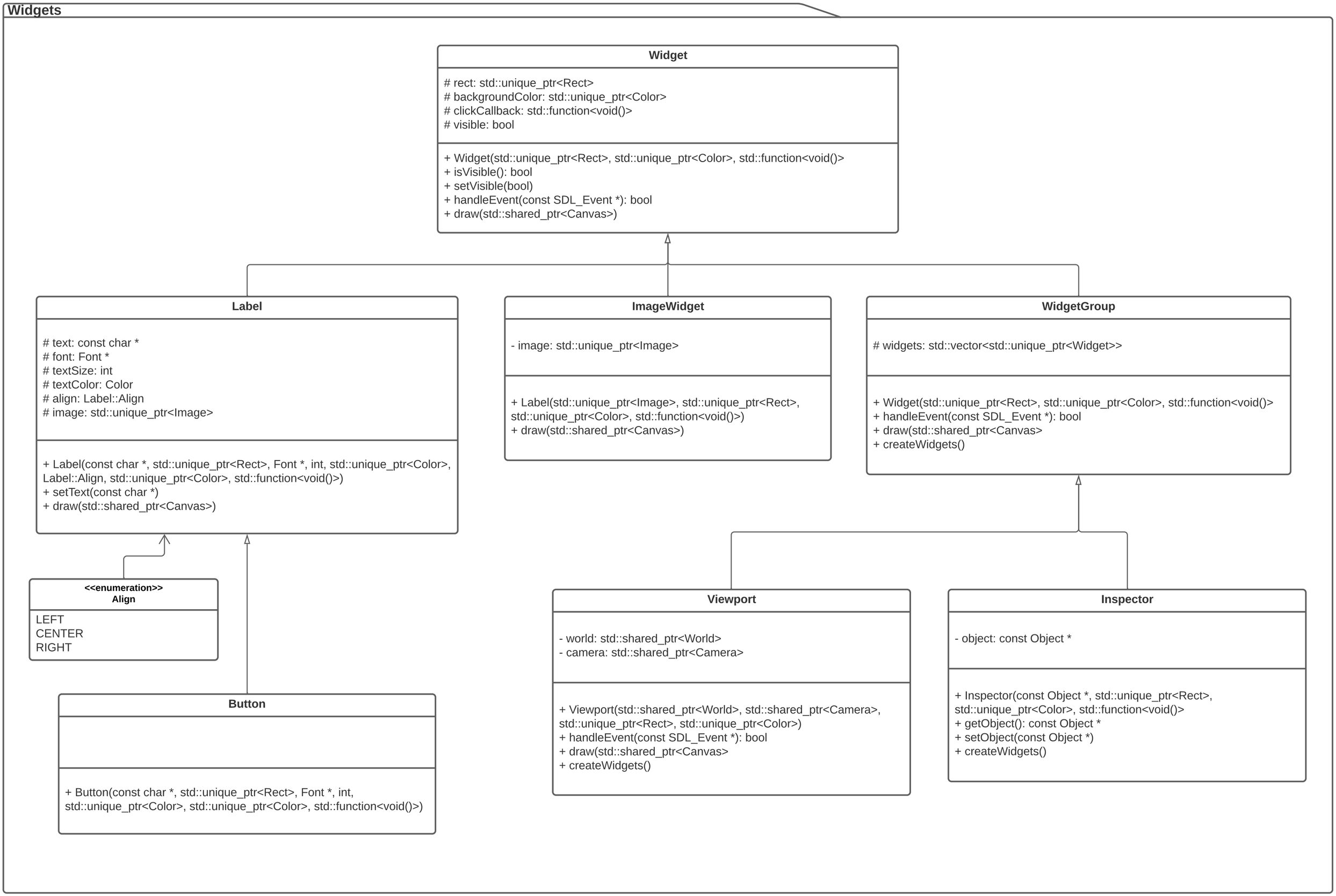
MenuPage

+ MenuPage()
+ createWidgets()

GamePage

- world: std::shared_ptr<World>
- camera: std::shared_ptr<Camera>

+ GamePage()
+ update(float)
+ createWidgets()


Widgets
Widget
rect: std::unique_ptr<Rect>
backgroundColor: std::unique_ptr<Color>
clickCallback: std::function<void()>
visible: bool
+ Widget(std::unique_ptr<Rect>, std::unique_ptr<Color>, std::function<void()>
+ isVisible(): bool
+ setVisible(bool)
+ handleEvent(const SDL_Event *): bool
+ draw(std::shared_ptr<Canvas>)
Label
text: const char *
font: Font *
textSize: int
textColor: Color
align: Label::Align
image: std::unique_ptr<Image>
+ Label(const char *, std::unique_ptr<Rect>, Font *, int, std::unique_ptr<Color>, Label::Align, std::unique_ptr<Color>, std::function<void()>)
+ setText(const char *)
+ draw(std::shared_ptr<Canvas>)
ImageWidget
- image: std::unique_ptr<Image>
+ Label(std::unique_ptr<Image>, std::unique_ptr<Rect>, std::unique_ptr<Color>, std::function<void()>)
+ draw(std::shared_ptr<Canvas>)
WidgetGroup
widgets: std::vector<std::unique_ptr<Widget>>
+ Widget(std::unique_ptr<Rect>, std::unique_ptr<Color>, std::function<void()>
+ handleEvent(const SDL_Event *): bool
+ draw(std::shared_ptr<Canvas>
+ createWidgets()
<<enumeration>>
Align
LEFT
CENTER
RIGHT
Button
+ Button(const char *, std::unique_ptr<Rect>, Font *, int, std::unique_ptr<Color>, std::unique_ptr<Color>, std::function<void()>)
Viewport
- world: std::shared_ptr<World>
- camera: std::shared_ptr<Camera>
+ Viewport(std::shared_ptr<World>, std::shared_ptr<Camera>, std::unique_ptr<Rect>, std::unique_ptr<Color>)
+ handleEvent(const SDL_Event *): bool
+ draw(std::shared_ptr<Canvas>
+ createWidgets()
Inspector
- object: const Object *
+ Inspector(const Object *, std::unique_ptr<Rect>, std::unique_ptr<Color>, std::function<void()>
+ getObject(): const Object *
+ setObject(const Object *)
+ createWidgets()

Objects

Object

id: int
x: float
y: float

Object(int, float, float)
+ getId(): int
+ getX(): float
+ getY(): float
+ handleEvent(const SDL_Event *event): bool
+ update(float)
+ draw(std::shared_ptr<Canvas>, const Camera *)

Nature

- images: std::unique_pr<Image>[Nature::Type::size]
- names: const char *[Nature::Type::size]
- type: Nature::Type

+ Nature(int, Nature::Type, float, float)
+ getType(): Nature::Type
+ draw(std::shared_ptr<Canvas>, const Camera *)
+ loadImages()
+ getImage(Nature::Type): const Image *
+ getName(Nature::Type): const char *

<<enumeration>>
Type

BUSHES = 0
BERRIES
STONE
GOLD
BEECH
BEECH_SMALL
FIR
FIR_SMALL
TRUNK
TRUNK_SMALL
size

House

- images: std::unique_ptr<Image>[House::Type::size]
- names: const char *[House::Type::size]
- type: House::Type
- population: int

+ House(id, House::Type, float, float, int)
+ getType(): House::Type
+ getPopulation(): int
+ draw(std::shared_ptr<Canvas>, const Camera *)
+ loadImages()
+ getImage(House::Type): const Image *
+ getName(House::Type): const char *

<<enumeration>>
Type

HOUSE = 0
HOUSE_SMALL
BACKERY
SHOP
size

Vehicle

- stats: Vehicle::Stats[Vehicle::Type::size]
- images: std::unique_ptr<Image>[Vehicle::Color::size]
[Vehicle::Type::size]
- type: Vehicle::Type
- color: Vehicle::Color
- angle: float
- velocity: float
- acceleration: float
- driver: std::unique_ptr<Driver>

+ Vehicle(int, Vehicle::Type, float, float, Vehice::Color, float)
+ getType(): Vehicle::Type
+ getColor(): Vehicle:: Color
+ getAngle(): float
+ getVelocity(): float
+ getAcceleration(): float
+ getDriver(): const Driver *
+ setDriver(std::unique_ptr<Driver>)
+ crash()
+ update(float)
+ draw(std::shared_ptr<Canvas>, const Camera *)
+ getStats(Vehicle::Type): const Vehicle::Stats *
+ loadImages()
+ getImage(Vehicle::Type, Vehicle::Color): const Image *

Stats

+ name: const char *
+ width: int
+ height: int
+ weight: int
+ maxForwardVelocity:int
+ forwardAcceleration: int
+ maxBackwardVelocity: int
+ backwardAcceleration: int
+ turningSpeed: int

<<enumeration>>
Type

SPORT = 0
MINI
STANDARD
TRUCK
TESLA
MOTOR_CYCLE
size

<<enumeration>>
Color

BLACK = 0
BLUE
GREEN
RED
YELLOW
size

City

- name: const char *
- population: int

+ City(int, const char *, float, float, int)
+ getName(): const char *
+ getPopulation(): int
+ setPopulation(int)
+ randomName(Random *): const char *
+ draw(std::shared_ptr<Canvas>, const Camera *)

Terrain

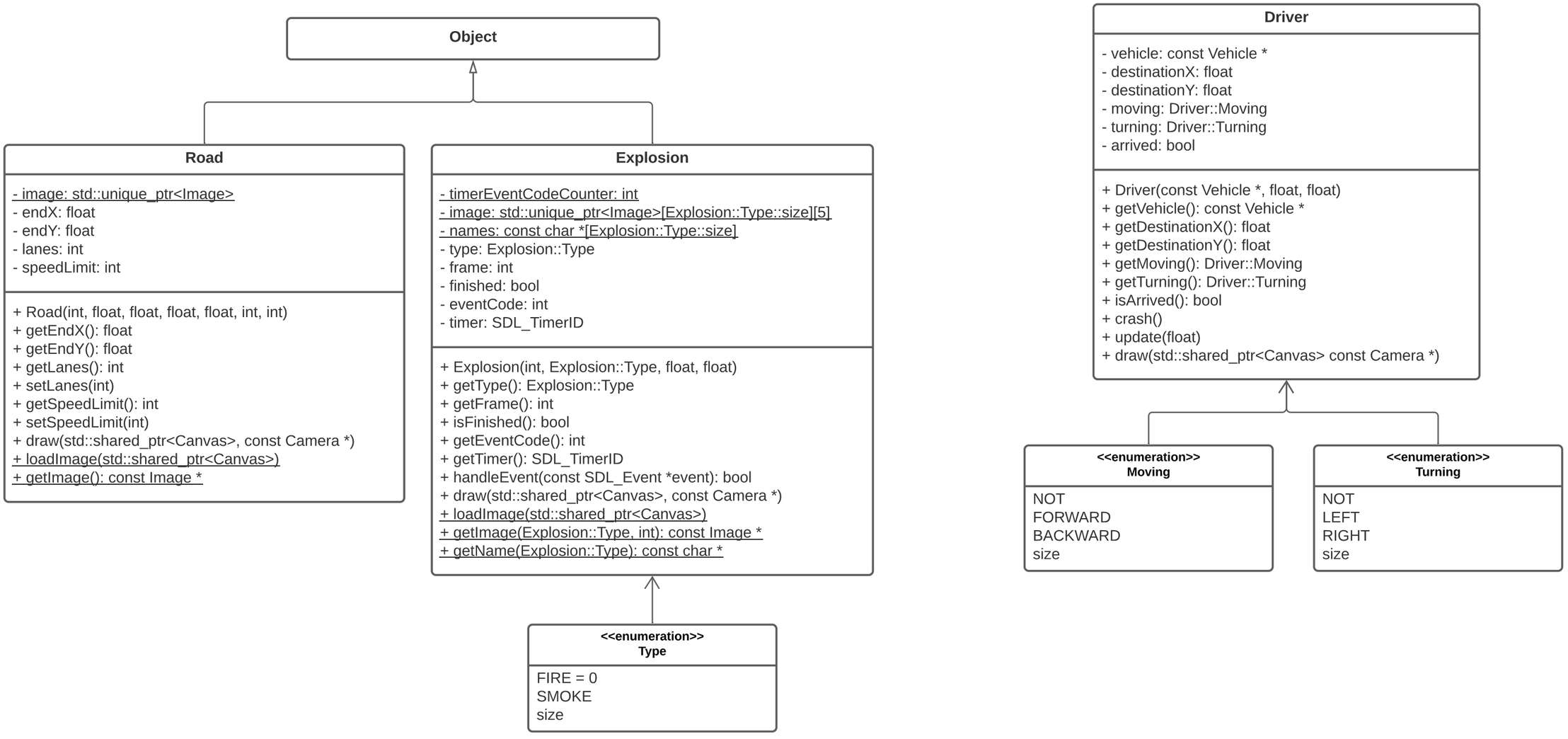
- images: std::unique_ptr<Image>[Terrain::Type::size
- names: const char *[Terrain::Type::size]

+ loadImages()
+ getImage(Terrain::Type): const Image *
+ getName(Terrain::Type): const char *

<<enumeration>>
Type

WATER_DEEP = 0
WATER
SAND1
SAND2
GRASS1
GRASS2
DIRT1
DIRT2
STONE1
STONE2
SNOW1
SNOW2
size

Objects


Object
Road
- image: std::unique_ptr<Image>
- endX: float
- endY: float
- lanes: int
- speedLimit: int
+ Road(int, float, float, float, float, int, int)
+ getEndX(): float
+ getEndY(): float
+ getLanes(): int
+ setLanes(int)
+ getSpeedLimit(): int
+ setSpeedLimit(int)
+ draw(std::shared_ptr<Canvas>, const Camera *)
+ loadImage(std::shared_ptr<Canvas>)
+ getImage(): const Image *
Explosion
- timerEventCodeCounter: int
- image: std::unique_ptr<Image>[Explosion::Type::size][5]
- names: const char *[Explosion::Type::size]
- type: Explosion::Type
- frame: int
- finished: bool
- eventCode: int
- timer: SDL_TimerID
+ Explosion(int, Explosion::Type, float, float)
+ getType(): Explosion::Type
+ getFrame(): int
+ isFinished(): bool
+ getEventCode(): int
+ getTimer(): SDL_TimerID
+ handleEvent(const SDL_Event *event): bool
+ draw(std::shared_ptr<Canvas>, const Camera *)
+ loadImage(std::shared_ptr<Canvas>)
+ getImage(Explosion::Type, int): const Image *
+ getName(Explosion::Type): const char *
<<enumeration>>
Type
FIRE = 0
SMOKE
size
Driver
- vehicle: const Vehicle *
- destinationX: float
- destinationY: float
- moving: Driver::Moving
- turning: Driver::Turning
- arrived: bool
+ Driver(const Vehicle *, float, float)
+ getVehicle(): const Vehicle *
+ getDestinationX(): float
+ getDestinationY(): float
+ getMoving(): Driver::Moving
+ getTurning(): Driver::Turning
+ isArrived(): bool
+ crash()
+ update(float)
+ draw(std::shared_ptr<Canvas> const Camera *)
<<enumeration>>
Moving
NOT
FORWARD
BACKWARD
size
<<enumeration>>
Turning
NOT
LEFT
RIGHT
size