MapServer UML

model

Map
- header: Header
- content: String

getHeader(void): String
allowCo(Map): boolean
getContent(void): String

Header
- name: String
- mapname: String

+ getName(void): String
+ getMapname(void): String
+ equals(object): boolean

User
- username: String
- salt: String
- hashedPassword: String

getUsername(void): String
getSalt(void): String
getHashedPassword(void): String

UserStatus
- username: String
- bombsKilled: Integer
- levelStar: List<Integer>
- maxLevel: Integer
- enemyKilled: Integer
- treasureCollected: Integer

getEnemyKilled(void): Integer
getLevelStar(void): List<Integer>
getBombsKilled(void): Integer
getMaxLevel(void): Integer
getTreasureCollected(void): Integer
getUsername(void): String

controller

mapController
+ NO_LOGIN: String
+ NO_VAR: String
+ DUP_MAP: String
+ MAP_SUCC: String
+ MAP_FAIL: String
+ getAllMaps: (Request, Response) -> Route
+ getOne: (Request, Response) -> Route
+ addNewMap: (Request, Response) -> Route

loginController
+ LOGIN_FAILED: String
+ LOGIN_SUCC: String
+ ALREADY_LOGGED_IN: String
+ LOGOUT_NO_SUCH_USER: String
+ LOGOUT_NOT_LOGGED_IN: String
+ LOGOUT_SUCC: String
+ REGISTER_FAIL: String
+ REGISTER_SUCC: String
+ handleLoginPost: (Request, Response) -> Route
+ handleLogoutPost: (Request, Response) -> Route
+ handleRegisterPost: (Request, Response) -> Route

+ userIsLoggedIn(Request): boolean

userController
+ updateStatus: (Request, Response) -> Route

+ authenticate(String): boolean
+ hasUser(String): Boolean
+ hasUser(String, String): boolean
+ addUser(String, String): void

Application
+ main(String[] args): void
- InitServer(void): void
- forceStop(String): void

util

JsonUtil
+ onlyUsername: ExclusionStrategy()

+ headerToJson(object): String
+ headerJson(void): ResponseTransformer
+ mapToJson(object): String
+ mapJson(void): ResponseTransformer

MapReader
- PATH: String
- files: ArrayList<File>

+ method(type): type
+ getHeader(void): List<Header>
+ parseHeader(File): Header
+ getMap(String, String): Map
- parseMap(File): Map
+ addMap(String, String, String): boolean
- thereIs(Map): Map
- checkMap(Map, Map): boolean

Path

Web
+ Path: String
+ Web: String
+ LOGIN: String
+ LOGOUT: String
+ REGISTER: String
+ REQUESTALL: String
+ REQUESTONE: String
+ UPLOAD: String
+ UPDATE: String

RequestUtil
+ getSessionCurrentUser(Request): String
+ getUploadedMap(Request): String
+ getQueryPassword(Request): String
+ getWantedUser(Request): String
+ getWantedMap(Request): String
+ getQueryUsername(Request): String
+ getUpdateName(Request): String
+ getUploadedMapName(Request): String
+ getenemyKilled(Request): Integer
+ gettreasureCollected(Request): Integer
+ hasmaxLevel(Request): Integer
+ getbombsKilled(Request): Integer
+ getLevelStar(Request): List<Integer>

addNewMap
+ mapName: String
+ mapContent: String

UserReader
- PATH1: String
- PATH2: String
- gson: Gson

UpdateUser(Request): void
hasUser(String): boolean
hasUser(String, String): boolean
addUser(String, String): void

userContent
+ username: String
+ password: String

+ getPassword(void): String
+ userContent(String, String)
+ getUsername(void): String

Door
+ field: type
+ method(type): type

Pit
+ field: type
+ method(type): type

Switch
+ field: type
+ method(type): type

Ground
+ field: type
+ method(type): type

Exit
+ field: type
+ method(type): type

Terrain
+ field: type
+ method(type): type

Wall
+ field: type
+ method(type): type

HeatPlate
+ field: type
+ method(type): type

Arrow
+ field: type
+ method(type): type

Treasure
+ field: type
+ method(type): type

Bomb
+ field: type
+ method(type): type

Key
+ field: type
+ method(type): type

Sword
+ field: type
+ method(type): type

Pickup
+ field: type
+ method(type): type

HoverPotion
+ field: type
+ method(type): type

BombPotion
+ field: type
+ method(type): type

InvincibilityPotion
+ field: type
+ method(type): type

Entity
+DEFAULT_SIZE: double
symbol: char
#pos: Vec2i
#level: Level
#view: ViewComponent
+ sprite: SpriteView
+ soundManager: SoundManager

+ onLevelRescaled(void): void
+ onCreated(void): void
+ onRemovedFromLevel(void):void
+ onExploded(void): void
+ onTurnUpdate(void): void
+ destroy(void): void
+ getView(void): Node
+ setPos(int, int): void
+ setPos(Vec2i): void
+ getGridPos(void): Vec2i
+ getX(void): int
+ getY(void): int
+ getWorldPos(void): Vec2d
+ getSymbol(void): char
+ getSprite(void): SpriteView
+ isPassableFor(Entity): boolean
+ isPassableForProp(Prop): boolean
+ isPassableForEnemy(Enemy): boolean
+ isPassableForAvatar(Avatar): boolean
+ canStackFor(Entity): boolean
+ canStackForTerrain(Terrain): boolean
+ canStackForProp(Prop): boolean
+ canStackForPickup(Pickup): boolean
+ canStackForEnemy(Enemy): boolean
+ canStackForAvatar(Avatar): boolean
+ onEnterBy(Entity):boolean
+ onEnterByProp(Prop): boolean
+ onEnterByEnemy(Enemy): boolean
+ onEnterByAvatar(Avatar): boolean
+ onLeaveBy(Entity): boolean
+ onLeaveByProp(Prop): boolean
+ onLeaveByEnemy(Enemy): boolean
+ onLeaveByAvatar(Avatar): boolean
+ getMetaData(void): String
+ getImage(String): Image
+ size(void): double

Prop
+ field: type
+ method(type): type

FlyingArrow
+ field: type
+ method(type): type

LitBomb
+ field: type
+ method(type): type

IceBlock
+ field: type
+ method(type): type

Boulder
+ field: type
+ method(type): type

Avatar
+ field: type
+ method(type): type

Hound
+ field: type
+ method(type): type

Hunter
+ field: type
+ method(type): type

Enemy
+ field: type
+ method(type): type

Coward
+ field: type
+ method(type): type

Strategist
+ field: type
+ method(type): type

EnemyCommander
+ field: type
+ method(type): type

HoundBehaviour
+ field: type
+ method(type): type

HunterBehaviour
+ field: type
+ method(type): type

AIBehaviour
+ field: type
+ method(type): type

CowardBehaviour
+ field: type
+ method(type): type

StrategistBehaviour
+ field: type
+ method(type): type

LevelBuilder

+ field: type

+ method(type): type

MapLoader

+ field: type

+ method(type): type

DraftBuilder

+ field: type

+ method(type): type

LayerBuilder

+ field: type

+ method(type): type

EntityFactory

+ field: type

+ method(type): type

EntityLayer

+ field: type

+ method(type): type

TerrainLayer

+ field: type

+ method(type): type

SingletonLayer

+ field: type

+ method(type): type

HashMapLayer

+ field: type

+ method(type): type

ImageView

Extends

SpriteView

- states HashMap<String, Image>
- offsets HashMap<String, Vec2d>
- scaleX HashMap<String, Double>
- scaleY HashMap<String, Double>
- anime HashMap<String, SpriteAnimation>

+ addState(String, Image): void
+ addState(String, Image, Vec2d, double, double): void
+ setState(String): void
+ addAnime(String, SpriteAnimation): void
+ playAnime(String, EventHandler<ActionEvent>):void
+ playAnimation(String, Vec2i, EventHandler<ActionEver
+ directionParser(Vec2i):String
+ magnifyScales(double, double): void

Transition

Extends

1

SpriteAnimation

- imageView: ImageView
- states: ArrayList<Image>
- offsets:ArrayList<Vec2d>
- lastIndex: int
- initialOffset: Vec2i
- scaleX: double
-scaleY: double
- beforePlay: Runnable
- afterPlay: Runnable

+ addState(String, Image): void
+ addState(String, Image, Vec2d): void
+ setState(int): void
+ alignToUp(double,int): void
+ alignToDown(double,int): void
+ alignToLeft(double,int): void
+ alignToRight(double,int): void
+ alignOffset(Vec2d): void
+ play(EventHandler<ActionEvent>): void

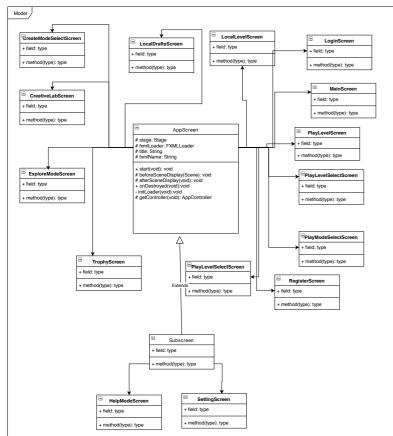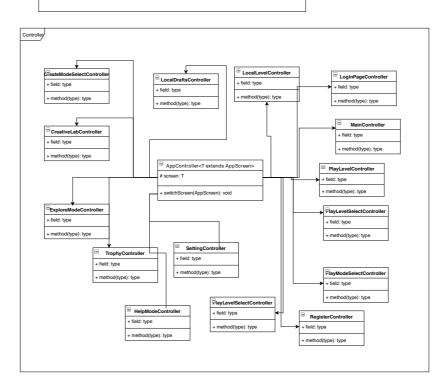| **SoundManager** |
| --- |
| - <u>soundManagerInstance: SoundManager</u>
- soundPool: ExecutorService
- soundEffectsMap: Map<String, AudioClip>
- FXVolume: double
- BGMPlayer: MediaPlayer
- BGMMap: Map<String, Media>
- BGMVolume: double |
| + getInstance(int): SoundManager
+ loadSoundEffect(String, String): void
+ loadMusic(String, String): void
+ playSoundEffect(String): void
+ playBGM(String): void
+ shutDown(void): void
+ getBGMVolume(void): double
+ getFXVolume(void): double
+ setBGMVolume(double): void
+ set FXVolume(double): void |

Model

CreateModeSelectScreen
+ field: type
+ method(type): type

LocalDraftsScreen
+ field: type
+ method(type): type

LocalLevelScreen
+ field: type
+ method(type): type

LoginScreen
+ field: type
+ method(type): type

CreativeLabScreen
+ field: type
+ method(type): type

MainScreen
+ field: type
+ method(type): type

AppScreen
stage: Stage
fxmlLoader: FXMLLoader
title: String
fxmlName: String
+ start(void): void
# beforeSceneDisplay(Scene): void
# afterSceneDisplay(void): void
# onDestroyed(void):void
- initLoader(void):void
getController(void): AppController

PlayLevelScreen
+ field: type
+ method(type): type

ExploreModeScreen
+ field: type
+ method(type): type

PlayLevelSelectScreen
+ field: type
+ method(type): type

PlayModeSelectScreen
+ field: type
+ method(type): type

TrophyScreen
+ field: type
+ method(type): type

PlayLevelSelectScreen
+ field: type
+ method(type): type

Extends

RegisterScreen
+ field: type
+ method(type): type

Subscreen
+ field: type
+ method(type): type

HelpModeScreen
+ field: type
+ method(type): type

SettingScreen
+ field: type
+ method(type): type

Main
- WIDTH: int
- HEIGHT: int
- CONFIG_FILENAME: String
+ soundManager: SoundManager
+ gameConfig: GameConfig
+ currClient: Client
+ primaryStage: Stage
+ achievementSystem: AchievementSystem

Engine

AlertHelper
+ showAlert(Alert, alertType, String, String): void

Input
+ startListening(void): void
+ stopListening(void): void
+ onKeyEvent(KeyEvent): void
+ addBinding(KeyCode, UserAction) void
+ removeBinding(KeyCode): void
+ update(void): void

<<Interface>>
Game
+ onStart(void): void
+ onUpdateBegin (void): void
+ onUpdate (void): void
+ onUpdateEnd (void): void
+ onStop (void): void

child

UserAction
+ wasActive: boolean
+ isActive: boolean
+ update(void): void
+ startAction(void): void
+ stopAction(void): void

parent

GameLoop
- NANO_PER_SEC: float-
SEC_PER_TICK: float
- MAX_TICKS_PER_FRAME: float
- LAG_CAP: int
- prevTime: long
- lag: float
- secondsElapsed: float
- nFrames: int
- game: Game
- sReporter: Consumer<Integer>

Extends → **AnimationTimer**

Controller

CreateModeSelectController
+ field: type
+ method(type): type

LocalDraftsController
+ field: type
+ method(type): type

LocalLevelController
+ field: type
+ method(type): type

LoginPageController
+ field: type
+ method(type): type

CreativeLabController
+ field: type
+ method(type): type

MainController
+ field: type
+ method(type): type

AppController<T extends AppScreen>
screen: T
+ switchScreen(AppScreen): void

PlayLevelController
+ field: type
+ method(type): type

ExploreModeController
+ field: type
+ method(type): type

PlayLevelSelectController
+ field: type
+ method(type): type

TrophyController
+ field: type
+ method(type): type

SettingController
+ field: type
+ method(type): type

PlayModeSelectController
+ field: type
+ method(type): type

HelpModeController
+ field: type
+ method(type): type

PlayLevelSelectController
+ field: type
+ method(type): type

RegisterController
+ field: type
+ method(type): type

Sprite

Entities

Algorithms

GameScheduler

+ field: type
+ method(type): type

App

Math

MapLoading

Client

Level

- view: ViewComponent
- size: double
- nRows: int
- nCols: int
- name: String
- isCreateMode: Boolean
- layers: ArrayList<EntityLayer>
- terrains: TerrainLayer
- props: HashMapLayer<Prop>
- pickups: HashMapLayer<Pickup>
- enemies: HashMapLayer<Enemy>
- avatarLayer: SingletonLayer<Avatar>
- eventBus: EventBus
- externalBuses: ArrayList<EventBus>
- objectiveSystem: ObjectiveSystem
- background: ImageView
+ rescale(double): void
+ getName(void): String
+ setName(String): void
+ getTerrain(Vec2i): Terrain
+ getProp(Vec2i): Prop
+ getPickup(Vec2i): Pickup
+ getEnemy(Vec2i): Enemy
+ getAvatar(Vec2i): Avatar
+ setTerrain(Vec2i, Terrain): void
+ setProp(Vec2i, Prop): void
+ setPickup(Vec2i, Pickup): void
+ setEnemy(Vec2i, Enemy): void
+ setAvatar(Vec2i, Avatar): void
+ addTerrain(Vec2i, Terrain): void
+ addProp(Vec2i, Prop): void
+ addPickup(Vec2i, Pickup): void
+ addEnemy(Vec2i, Enemy): void
+ addAvatar(Vec2i, Avatar): void
+ removeAllAt(Vec2i, boolean): void
+ removeTerrain(Vec2i): void
+ removeProp(Vec2i): void
+ removePickup(Vec2i): void
+ removeEnemy(Vec2i): void
+ removeAvatar(Vec2i): void
+ moveTerrain(Vec2i, Terrain): void
+ moveProp(Vec2i, Prop): void
+ movePickup(Vec2i, Pickup): void
+ moveEnemy(Vec2i, Enemy): void
+ moveAvatar(Vec2i, Avatar): void
+ notifyOnEnterByProp(Vec2i, Prop): void
+ notifyOnEnterByEnemy(Vec2i, Enemy): void
+ notifyOnEnterByAvatar(Vec2i, Avatar): void
+ notifyOnLeaveByProp(Vec2i, Prop): void
+ notifyOnLeaveByEnemy(Vec2i, Enemy): void
+ notifyOnLeaveByAvatar(Vec2i, Avatar): void
+ forEachEntityAt(Vec2i, Consumer<Entity>): void
+ hasTerrain(Vec2i): boolean
+ hasProp(Vec2i): boolean
+ hasPickup(Vec2i): boolean
+ hasEnemy(Vec2i): boolean
+ hasAvatar(Vec2i): boolean
+ canPlaceTerrain(Vec2i, Terrain): boolean
+ canPlaceProp(Vec2i, Prop): boolean
+ canPlacePickup(Vec2i, Pickup): boolean
+ canPlaceEnemy(Vec2i, Enemy):boolean
+ canPlaceAvatar(Vec2i, Avatar):boolean
+ canReplaceTerrain(Vec2i, Terrain): boolean
+ canReplaceProp(Vec2i, Prop): boolean
+ canReplacePickup(Vec2i, Pickup): boolean
+ canReplaceEnemy(Vec2i, Enemy):boolean
+ canReplaceAvatar(Vec2i, Avatar):boolean
+ isPassableForProp(Vec2i, Prop): boolean
+ isPassableForPickup(Vec2i, Pickup): boolean
+ isPassableForEnemy(Vec2i, Enemy):boolean
+ isPassableForAvatar(Vec2i, Avatar):boolean
+ getNRows(void): int
+ getNCols(void): int
+ getHeight(void): double
+ getWidth(void): double
+ isValidGridPos(Vec2i): boolean
+ gridPosToWorldPosCentre(Vec2i): Vec2d
+ resize(int,int): void
+ getTerrainIterator(void): Iterator<Terrain>
+ getPropIterator(void): Iterator<Prop>
+ getPickupIterator(void): Iterator<Pickup>
+ getEnemyIterator(void): Iterator<Enemy>
+ getEnemies(void): ArrayList<Enemy>
+ getEntitiesAt(Vec2i): Iterator<Entity>
+ displayLevel(void): void
+ postEvent(Event): void
+ addBus(Event): EventBus
+ addEventHandler(EventType, EventHandler): Event
+ removeEventHandler(EventType, EventHandler): void
+ addObjective(Objective): void
+ checkAchievedAllObjectives(void): boolean
+ isCreateMode(void): boolean
+ setCreateMode(boolean): void
+ listObjectives(void):String
+ clearObjectives(void): void
+ getSize(void): double
+ toFile(String, String): File
+ containsExit(void): boolean

Trigger

Content

Events

LevelPlayer

+ field: type
+ method(type): type

PlayModeUILocator

+ field: type
+ method(type): type