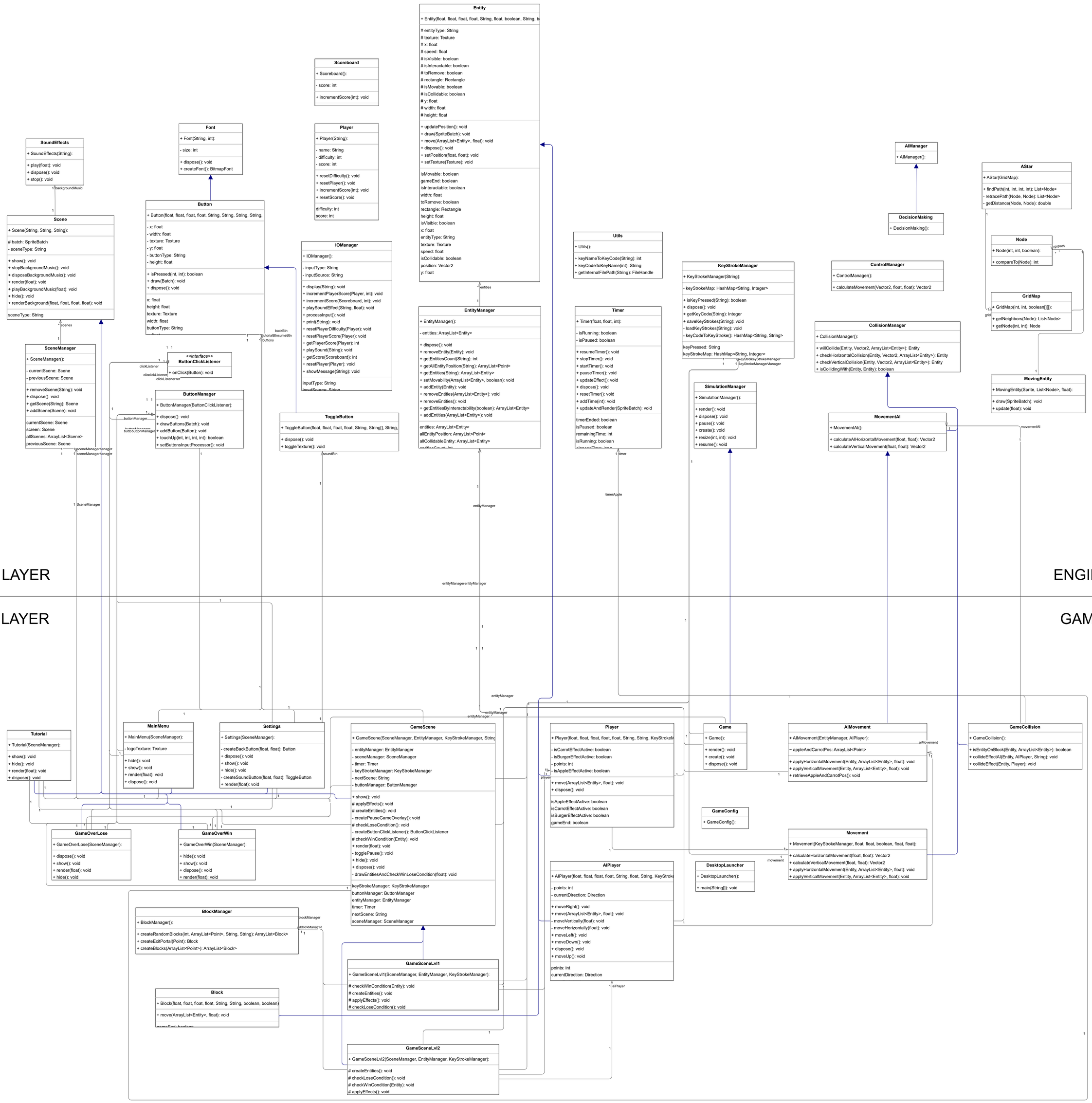
ENGINE LAYER
ENGINE LAYER
GAME LAYER
GAME LAYER
Entity
+ Entity(float, float, float, float, String, float, boolean, String, b
# entityType: String
# texture: Texture
# x: float
# speed: float
# isVisible: boolean
# isInteractable: boolean
# toRemove: boolean
# rectangle: Rectangle
# isMovable: boolean
# isCollidable: boolean
# y: float
# width: float
# height: float
+ updatePosition(): void
+ draw(SpriteBatch): void
+ move(ArrayList<Entity>, float): void
+ dispose(): void
+ setPosition(float, float): void
+ setTexture(Texture): void
isMovable: boolean
gameEnd: boolean
isInteractable: boolean
width: float
toRemove: boolean
rectangle: Rectangle
height: float
isVisible: boolean
x: float
entityType: String
texture: Texture
speed: float
isCollidable: boolean
position: Vector2
y: float
Scoreboard
+ Scoreboard():
- score: int
+ incrementScore(int): void
Font
+ Font(String, int):
- size: int
+ dispose(): void
+ createFont(): BitmapFont
Player
+ Player(String):
- name: String
- difficulty: int
- score: int
+ resetDifficulty(): void
+ resetPlayer(): void
+ incrementScore(int): void
+ resetScore(): void
difficulty: int
score: int
SoundEffects
+ SoundEffects(String):
+ play(float): void
+ dispose(): void
+ stop(): void
Button
+ Button(float, float, float, float, String, String, String, String,
- x: float
- width: float
- texture: Texture
- y: float
- buttonType: String
- height: float
+ isPressed(int, int): boolean
+ draw(Batch): void
+ dispose(): void
x: float
height: float
texture: Texture
width: float
buttonType: String
IOManager
+ IOManager():
- inputType: String
- inputSource: String
+ display(String): void
+ incrementPlayerScore(Player, int): void
+ incrementScore(Scoreboard, int): void
+ playSoundEffect(String, float): void
+ processInput(): void
+ print(String): void
+ resetPlayerDifficulty(Player): void
+ resetPlayerScore(Player): void
+ getPlayerScore(Player): int
+ playSound(String): void
+ getScore(Scoreboard): int
+ resetPlayer(Player): void
+ showMessage(String): void
inputType: String
Utils
+ Utils():
+ keyNameToKeyCode(String): int
+ keyCodeToKeyName(int): String
+ getInternalFilePath(String): FileHandle
AIManager
+ AIManager():
AStar
+ AStar(GridMap):
+ findPath(int, int, int, int): List<Node>
- retracePath(Node, Node): List<Node>
- getDistance(Node, Node): double
DecisionMaking
+ DecisionMaking():
Node
+ Node(int, int, boolean):
+ compareTo(Node): int
GridMap
+ GridMap(int, int, boolean[][]):
+ getNeighbors(Node): List<Node>
+ getNode(int, int): Node
Scene
+ Scene(String, String, String):
# batch: SpriteBatch
- sceneType: String
+ show(): void
+ stopBackgroundMusic(): void
+ disposeBackgroundMusic(): void
+ render(float): void
+ playBackgroundMusic(float): void
+ hide(): void
+ renderBackground(float, float, float, float): void
sceneType: String
EntityManager
+ EntityManager():
- entities: ArrayList<Entity>
+ dispose(): void
+ removeEntity(Entity): void
+ getEntitiesCount(String): int
+ getAllEntityPosition(String): ArrayList<Point>
+ getEntities(String): ArrayList<Entity>
+ setMovability(ArrayList<Entity>, boolean): void
+ addEntity(Entity): void
+ removeEntities(ArrayList<Entity>): void
+ removeEntities(): void
+ getEntitiesByInteractability(boolean): ArrayList<Entity>
+ addEntities(ArrayList<Entity>): void
entities: ArrayList<Entity>
allEntityPosition: ArrayList<Point>
allCollidableEntity: ArrayList<Entity>
Timer
+ Timer(float, float, int):
- isRunning: boolean
- isPaused: boolean
+ resumeTimer(): void
+ stopTimer(): void
+ startTimer(): void
+ pauseTimer(): void
+ updateEffect(): void
+ dispose(): void
+ resetTimer(): void
+ addTime(int): void
+ updateAndRender(SpriteBatch): void
timerEnded: boolean
isPaused: boolean
remainingTime: int
isRunning: boolean
KeyStrokeManager
+ KeyStrokeManager(String):
- keyStrokeMap: HashMap<String, Integer>
+ isKeyPressed(String): boolean
+ dispose(): void
+ getKeyCode(String): Integer
+ saveKeyStrokes(String): void
- loadKeyStrokes(String): void
- keyCodeToKeyStroke(): HashMap<String, String>
keyPressed: String
keyStrokeMap: HashMap<String, Integer>
ControlManager
+ ControlManager():
+ calculateMovement(Vector2, float, float): Vector2
CollisionManager
+ CollisionManager():
+ willCollide(Entity, Vector2, ArrayList<Entity>): Entity
+ checkHorizontalCollision(Entity, Vector2, ArrayList<Entity>): Entity
+ checkVerticalCollision(Entity, Vector2, ArrayList<Entity>): Entity
+ isCollidingWith(Entity, Entity): boolean
MovingEntity
+ MovingEntity(Sprite, List<Node>, float):
+ draw(SpriteBatch): void
+ update(float): void
SceneManager
+ SceneManager():
- currentScene: Scene
- previousScene: Scene
+ removeScene(String): void
+ dispose(): void
+ getScene(String): Scene
+ addScene(Scene): void
currentScene: Scene
screen: Scene
allScenes: ArrayList<Scene>
previousScene: Scene
<<interface>>
ButtonClickListener
+ onClick(Button): void
ButtonManager
+ ButtonManager(ButtonClickListener):
+ dispose(): void
+ drawButtons(Batch): void
+ addButton(Button): void
+ touchUp(int, int, int, int): boolean
+ setButtonsInputProcessor(): void
ToggleButton
+ ToggleButton(float, float, float, float, String, String[], String,
+ dispose(): void
+ toggleTexture(): void
SimulationManager
+ SimulationManager():
+ render(): void
+ dispose(): void
+ pause(): void
+ create(): void
+ resize(int, int): void
+ resume(): void
MovementAI
+ MovementAI():
+ calculateAIHorizontalMovement(float, float): Vector2
+ calculateAIVerticalMovement(float, float): Vector2
Tutorial
+ Tutorial(SceneManager):
+ show(): void
+ hide(): void
+ render(float): void
+ dispose(): void
MainMenu
+ MainMenu(SceneManager):
- logoTexture: Texture
+ hide(): void
+ show(): void
+ render(float): void
+ dispose(): void
Settings
+ Settings(SceneManager):
- createBackButton(float, float): Button
+ dispose(): void
+ show(): void
+ hide(): void
- createSoundButton(float, float): ToggleButton
+ render(float): void
GameScene
+ GameScene(SceneManager, EntityManager, KeyStrokeManager, String
- entityManager: EntityManager
- sceneManager: SceneManager
- timer: Timer
- keyStrokeManager: KeyStrokeManager
- nextScene: String
- buttonManager: ButtonManager
+ show(): void
# applyEffects(): void
# createEntities(): void
- createPauseGameOverlay(): void
# checkLoseCondition(): void
- createButtonClickListener(): ButtonClickListener
# checkWinCondition(Entity): void
+ render(float): void
- togglePause(): void
+ hide(): void
+ dispose(): void
- drawEntitiesAndCheckWinLoseCondition(float): void
keyStrokeManager: KeyStrokeManager
buttonManager: ButtonManager
entityManager: EntityManager
timer: Timer
nextScene: String
sceneManager: SceneManager
Player
+ Player(float, float, float, float, float, String, String, KeyStrokeM
- isCarrotEffectActive: boolean
- isBurgerEffectActive: boolean
- points: int
- isAppleEffectActive: boolean
+ move(ArrayList<Entity>, float): void
+ dispose(): void
isAppleEffectActive: boolean
isCarrotEffectActive: boolean
isBurgerEffectActive: boolean
gameEnd: boolean
Game
+ Game():
+ render(): void
+ create(): void
+ dispose(): void
AIMovement
+ AIMovement(EntityManager, AIPlayer):
~ appleAndCarrotPos: ArrayList<Point>
+ applyHorizontalMovement(Entity, ArrayList<Entity>, float): void
+ applyVerticalMovement(Entity, ArrayList<Entity>, float): void
+ retrieveAppleAndCarrotPos(): void
GameCollision
+ GameCollision():
+ isEntityOnBlock(Entity, ArrayList<Entity>): boolean
+ collideEffectAI(Entity, AIPlayer, String): void
+ collideEffect(Entity, Player): void
GameOverLose
+ GameOverLose(SceneManager):
+ dispose(): void
+ show(): void
+ render(float): void
+ hide(): void
GameOverWin
+ GameOverWin(SceneManager):
+ hide(): void
+ show(): void
+ dispose(): void
+ render(float): void
GameConfig
+ GameConfig():
Movement
+ Movement(KeyStrokeManager, float, float, boolean, float, float):
+ calculateHorizontalMovement(float, float): Vector2
+ calculateVerticalMovement(float, float): Vector2
+ applyHorizontalMovement(Entity, ArrayList<Entity>, float): void
+ applyVerticalMovement(Entity, ArrayList<Entity>, float): void
AIPlayer
+ AIPlayer(float, float, float, float, String, float, String, KeyStroke
- points: int
- currentDirection: Direction
+ moveRight(): void
+ move(ArrayList<Entity>, float): void
- moveVertically(float): void
- moveHorizontally(float): void
+ moveLeft(): void
+ moveDown(): void
+ dispose(): void
+ moveUp(): void
points: int
currentDirection: Direction
DesktopLauncher
+ DesktopLauncher():
+ main(String[]): void
BlockManager
+ BlockManager():
+ createRandomBlocks(int, ArrayList<Point>, String, String): ArrayList<Block>
+ createExitPortal(Point): Block
+ createBlocks(ArrayList<Point>): ArrayList<Block>
GameSceneLvl1
+ GameSceneLvl1(SceneManager, EntityManager, KeyStrokeManager):
# checkWinCondition(Entity): void
# createEntities(): void
# applyEffects(): void
# checkLoseCondition(): void
Block
+ Block(float, float, float, float, String, String, boolean, boolean)
+ move(ArrayList<Entity>, float): void
GameSceneLvl2
+ GameSceneLvl2(SceneManager, EntityManager, KeyStrokeManager):
# createEntities(): void
# checkLoseCondition(): void
# checkWinCondition(Entity): void
# applyEffects(): void