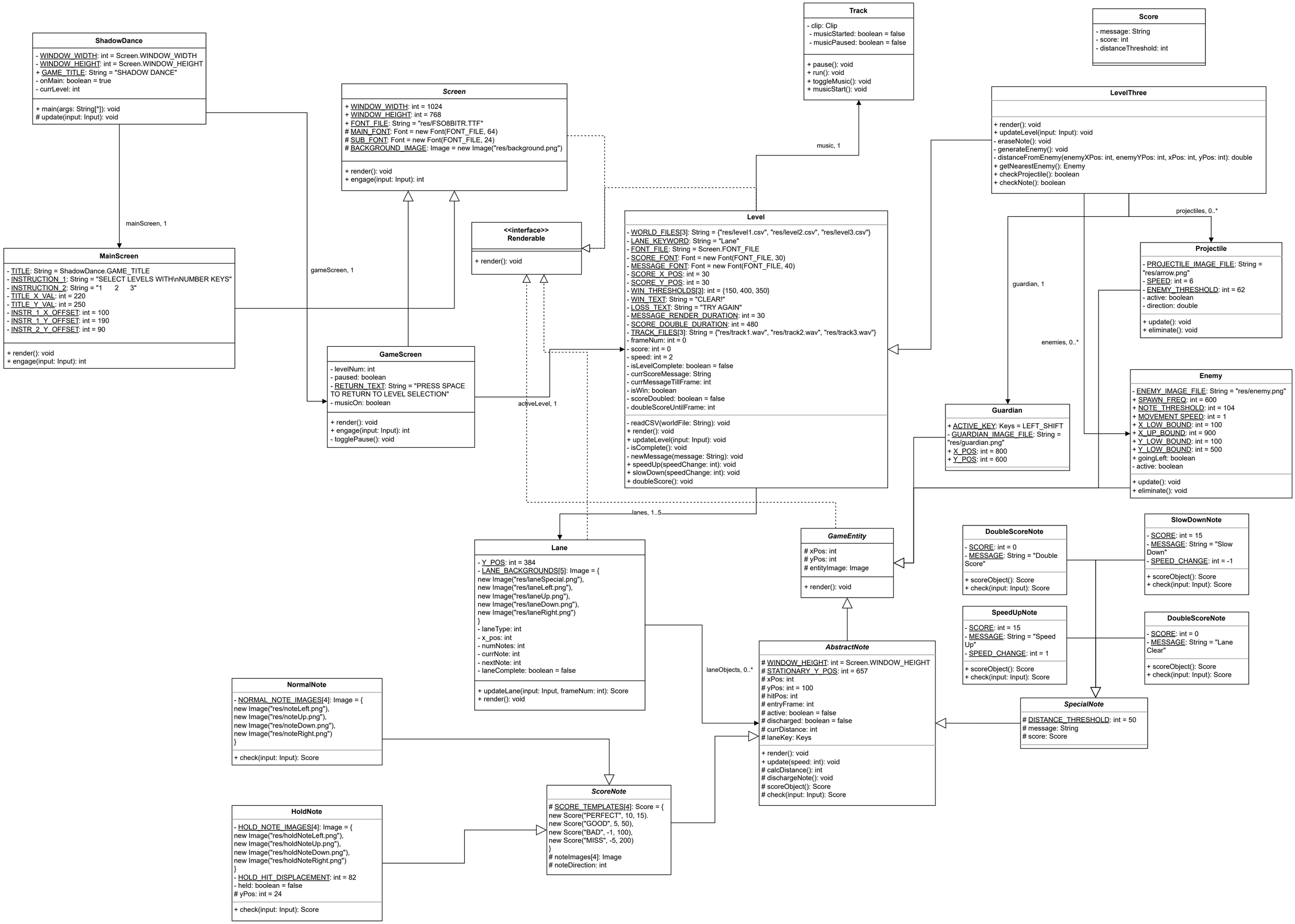
ShadowDance
- WINDOW_WIDTH: int = Screen.WINDOW_WIDTH
- WINDOW_HEIGHT: int = Screen.WINDOW_HEIGHT
+ GAME_TITLE: String = "SHADOW DANCE"
- onMain: boolean = true
- currLevel: int
+ main(args: String[*]): void
# update(input: Input): void
mainScreen, 1
gameScreen, 1
MainScreen
- TITLE: String = ShadowDance.GAME_TITLE
- INSTRUCTION_1: String = "SELECT LEVELS WITH\nNUMBER KEYS"
- INSTRUCTION_2: String = "1 2 3"
- TITLE_X_VAL: int = 220
- TITLE_Y_VAL: int = 250
- INSTR_1_X_OFFSET: int = 100
- INSTR_1_Y_OFFSET: int = 190
- INSTR_2_Y_OFFSET: int = 90
+ render(): void
+ engage(input: Input): int
Screen
+ WINDOW_WIDTH: int = 1024
+ WINDOW_HEIGHT: int = 768
+ FONT_FILE: String = "res/FSO8BITR.TTF"
# MAIN_FONT: Font = new Font(FONT_FILE, 64)
# SUB_FONT: Font = new Font(FONT_FILE, 24)
# BACKGROUND_IMAGE: Image = new Image("res/background.png")
+ render(): void
+ engage(input: Input): int
<<interface>>
Renderable
+ render(): void
GameScreen
- levelNum: int
- paused: boolean
- RETURN_TEXT: String = "PRESS SPACE TO RETURN TO LEVEL SELECTION"
- musicOn: boolean
+ render(): void
+ engage(input: Input): int
- togglePause(): void
activeLevel, 1
Track
- clip: Clip
- musicStarted: boolean = false
- musicPaused: boolean = false
+ pause(): void
+ run(): void
+ toggleMusic(): void
+ musicStart(): void
music, 1
Level
- WORLD_FILES[3]: String = {"res/level1.csv", "res/level2.csv", "res/level3.csv"}
- LANE_KEYWORD: String = "Lane"
- FONT_FILE: String = Screen.FONT_FILE
- SCORE_FONT: Font = new Font(FONT_FILE, 30)
- MESSAGE_FONT: Font = new Font(FONT_FILE, 40)
- SCORE_X_POS: int = 30
- SCORE_Y_POS: int = 30
- WIN_THRESHOLDS[3]: int = {150, 400, 350}
- WIN_TEXT: String = "CLEAR!"
- LOSS_TEXT: String = "TRY AGAIN"
- MESSAGE_RENDER_DURATION: int = 30
- SCORE_DOUBLE_DURATION: int = 480
- TRACK_FILES[3]: String = {"res/track1.wav", "res/track2.wav", "res/track3.wav"}
- frameNum: int = 0
- score: int = 0
- speed: int = 2
- isLevelComplete: boolean = false
- currScoreMessage: String
- currMessageTillFrame: int
- isWin: boolean
- scoreDoubled: boolean = false
- doubleScoreUntilFrame: int
- readCSV(worldFile: String): void
+ render(): void
+ updateLevel(input: Input): void
- isComplete(): void
- newMessage(message: String): void
+ speedUp(speedChange: int): void
+ slowDown(speedChange: int): void
+ doubleScore(): void
lanes, 1..5
Score
- message: String
- score: int
- distanceThreshold: int
LevelThree
+ render(): void
+ updateLevel(input: Input): void
- eraseNote(): void
- generateEnemy(): void
- distanceFromEnemy(enemyXPos: int, enemyYPos: int, xPos: int, yPos: int): double
+ getNearestEnemy(): Enemy
+ checkProjectile(): boolean
+ checkNote(): boolean
projectiles, 0..*
guardian, 1
enemies, 0..*
Projectile
- PROJECTILE_IMAGE_FILE: String = "res/arrow.png"
- SPEED: int = 6
- ENEMY_THRESHOLD: int = 62
- active: boolean
- direction: double
+ update(): void
+ eliminate(): void
Guardian
+ ACTIVE_KEY: Keys = LEFT_SHIFT
- GUARDIAN_IMAGE_FILE: String = "res/guardian.png"
+ X_POS: int = 800
+ Y_POS: int = 600
Enemy
- ENEMY_IMAGE_FILE: String = "res/enemy.png"
+ SPAWN_FREQ: int = 600
+ NOTE_THRESHOLD: int = 104
+ MOVEMENT SPEED: int = 1
+ X_LOW_BOUND: int = 100
+ X_UP_BOUND: int = 900
+ Y_LOW_BOUND: int = 100
+ Y_LOW_BOUND: int = 500
+ goingLeft: boolean
- active: boolean
+ update(): void
+ eliminate(): void
Lane
- Y_POS: int = 384
- LANE_BACKGROUNDS[5]: Image = {
new Image("res/laneSpecial.png"),
new Image("res/laneLeft.png"),
new Image("res/laneUp.png"),
new Image("res/laneDown.png"),
new Image("res/laneRight.png")
}
- laneType: int
- x_pos: int
- numNotes: int
- currNote: int
- nextNote: int
- laneComplete: boolean = false
+ updateLane(input: Input, frameNum: int): Score
+ render(): void
laneObjects, 0..*
GameEntity
xPos: int
yPos: int
entityImage: Image
+ render(): void
AbstractNote
WINDOW_HEIGHT: int = Screen.WINDOW_HEIGHT
STATIONARY_Y_POS: int = 657
xPos: int
yPos: int = 100
hitPos: int
entryFrame: int
active: boolean = false
discharged: boolean = false
currDistance: int
laneKey: Keys
+ render(): void
+ update(speed: int): void
# calcDistance(): int
# dischargeNote(): void
# scoreObject(): Score
# check(input: Input): Score
DoubleScoreNote
- SCORE: int = 0
- MESSAGE: String = "Double Score"
+ scoreObject(): Score
+ check(input: Input): Score
SlowDownNote
- SCORE: int = 15
- MESSAGE: String = "Slow Down"
- SPEED_CHANGE: int = -1
+ scoreObject(): Score
+ check(input: Input): Score
SpeedUpNote
- SCORE: int = 15
- MESSAGE: String = "Speed Up"
- SPEED_CHANGE: int = 1
+ scoreObject(): Score
+ check(input: Input): Score
DoubleScoreNote
- SCORE: int = 0
- MESSAGE: String = "Lane Clear"
+ scoreObject(): Score
+ check(input: Input): Score
SpecialNote
DISTANCE_THRESHOLD: int = 50
message: String
score: Score
NormalNote
- NORMAL_NOTE_IMAGES[4]: Image = {
new Image("res/noteLeft.png"),
new Image("res/noteUp.png"),
new Image("res/noteDown.png"),
new Image("res/noteRight.png")
}
+ check(input: Input): Score
ScoreNote
SCORE_TEMPLATES[4]: Score = {
new Score("PERFECT", 10, 15).
new Score("GOOD", 5, 50),
new Score("BAD", -1, 100),
new Score("MISS", -5, 200)
}
# noteImages[4]: Image
# noteDirection: int
HoldNote
- HOLD_NOTE_IMAGES[4]: Image = {
new Image("res/holdNoteLeft.png"),
new Image("res/holdNoteUp.png"),
new Image("res/holdNoteDown.png"),
new Image("res/holdNoteRight.png")
}
- HOLD_HIT_DISPLACEMENT: int = 82
- held: boolean = false
yPos: int = 24
+ check(input: Input): Score