Tetra's Adventure <<abstract>> Scene # maxX: number Tetra # maxY: number Mouse # backgroundImage: HTMLImageElement canvas: HTMLCanvasElement # mouse: Mouse -----# button: Button mouseListener: MouseListener constructor() currentLevel: Scene # continue: boolean <<abstract>> # mouseX: number Level constructor (canvas: HTMLCanvasElement) # mouseY: number processInput(): void update(elapsed: number): void # alien: Alien constructor (maxX: number, maxY: number) render(): void processInput(mouseListener: MouseListener): void # scoreItems: ScoreItem[] <<abstract>> getNextScene(): Scene | null # currentScore: number render(canvas: HTMLCanvasElement): void # currentLevel: number Alien # runButton: Button # hintButton: Button constructor (posX: number, posY: number) <<abstract>> processInput(mouseListener: MouseListener): void # portal: Button CanvasItem update(elapsed: number): void moveForward(): void # showHint: boolean # image: HTMLImageElement setPosition(posX: number, posY: number): void # poxX: number # hintlmage: HTMLlmageElement # poxY: number # positiveFeedback: HTMLImageElement Instruction SceneStart # negativeFeedback: HTMLImageElement getPosX(): number -----# displayFeedback: boolean getPosY(): number constructor (maxX: number, maxY: number) constructor (maxX: number, maxY: number) getWidth(): number getNextScene(): Scene | null getNextScene(): Scene | null getHeight(): number # inputButton: Button move(posX: number, posY: number): void # input: HTMLSelectElement Scoreltem isCollidingWithItem(item: CanvasItem): boolean # optionHint: HTMLOptionElement rotate(canvas: HTMLCanvasElement, degrees: - score: number # option01: HTMLOptionElement number): void # option02: HTMLOptionElement render(canvas: HTMLCanvasElement): void constructor(posX: number, posY: number, name: # option03: HTMLOptionElement SceneWin # option04: HTMLOptionElement IntroStory # option05: HTMLOptionElement ----- constructor (maxX: number, maxY: number) # option06: HTMLOptionElement constructor (maxX: number, maxY: number) getNextScene(): Scene | null # inputArray: string[] getNextScene(): Scene | null Button + constructor (maxX: number, maxY: number) + <<abstract>> processInput(mouseListener: MouseListener): void constructor(posX: number, posY: number, name: + <<abstract>> update(elapsed: number): void string) + <<abstract>> getNextScene(): Scene | null + <<abstract>> render(canvas: HTMLCanvasElement): void Credit SceneLose -----constructor (maxX: number, maxY: number) + constructor (maxX: number, maxY: number) getNextScene(): Scene | null getNextScene(): Scene | null Level1 Level2 Level3 _____ ________ _______ ---------------+ constructor (maxX: number, maxY: number) constructor (maxX: number, maxY: number) constructor (maxX: number, maxY: number) processInput(mouseListener: MouseListener): void processInput(mouseListener: MouseListener): void processInput(mouseListener: MouseListener): void update(elapsed: number): void update(elapsed: number): void update(elapsed: number): void getNextLevel(): Level | null getNextLevel(): Level | null getNextLevel(): Level | null render(canvas: HTMLCanvasElement): void render(canvas: HTMLCanvasElement): void render(canvas: HTMLCanvasElement): void Level4 Level5 Level6 ______ -----______ constructor (maxX: number, maxY: number) constructor (maxX: number, maxY: number) constructor (maxX: number, maxY: number) processInput(mouseListener: MouseListener): void processInput(mouseListener: MouseListener): void processInput(mouseListener: MouseListener): void update(elapsed: number): void update(elapsed: number): void update(elapsed: number): void getNextLevel(): Level | null getNextLevel(): Level | null getNextLevel(): Level | null render(canvas: HTMLCanvasElement): void render(canvas: HTMLCanvasElement): void render(canvas: HTMLCanvasElement): void