

GUI **Assumptions** UIElement* elementArray[MAX_ELEMENTS] General: All entities with a collider know the function Collider* GetCollider() const UIElement* elementDragged SDL Texture* atlas 1: We have all pointers necessary to all enemy related p2SString atlasFileName values 2:We have all pointers necessary to all particle related j1Gui(); virtual ~j1Gui() 3: We have all functions necessary to perform OnCollision bool Awake(pugi::xml_node&) (such as check direction, etc.) bool Start() bool PreUpdate() 4: We have an enum with all entity types bool PostUpdate() bool CleanUp() 5: Presence of non-specific logic-related bools, such as bool flip to know if we need to draw the texture flipped or UIElement* CreateButton(int x, int y, UIElement* father, j1Module* listeners[10], SDL_Rect* buttonIdle, SDL_Rect* buttonSelected, SDL_Rect* buttonPressed, bool dragable) 6: We have the enums of Explosion_Direction, Player_States and Player_Inputs UIElement* CreateUIWindow(int x, int y, UIElement* father, SDL_Rect* rect, bool dragable) 7: We have all variables "int" or "float" necessary to perform player functions (such as speed, time spent jumping, CD UIElement* CreateText(int x, int y, UIElement* father, from last explosion, etc.) _TTF_Font* font, p2SString &text, bool dragable) 8: We do have pointers to all animations and textures used UIElement* CreateScrollBar(int x, int y, UIElement* father, in player (walking animation, jumping animation, etc.) j1Module* listener, SDL_Rect* bar, SDL_Rect* thumbsIdle, SDL Rect* thumbsPressed) 9: We have a enum containing all types of UI_elements SDL_Texture* GetAtlas() const 10: We have some bools regarding logic: started, to_delete, void ListenerUI(UIElement* UI_element) void DragElement() void CheckFocusedElements() 11: We have an enum containing all possible button states **UI_Element Button** Assumption 9 Assumption 10 + 11 Assumption 10 SDL_Rect* button_idle SDL_Rect* button_selected SDL_Rect* my_box SDL_Rect* button_pressed iPoint position i1Module* listeners[10] iPoint local_position UI type type Button(int x, int y, UIElement* father, j1Module* listeners[10], SDL_Rect* buttonIdle, bool dragable SDL_Rect* buttonSelected, SDL_Rect* buttonPressed, bool dragable) bool focused ~Button() bool Start() SDL_Texture* texture bool Draw() UIElement* father void Speaker(j1Module* listener) void SpeakToAll() UIElement() void HandleInput(); UIElement(int x, int y, UIElement* father, bool dragable, UI_type type) virtual ~UIElement() virtual bool Start() TTF Font* font Tex bool Update() p2SString string virtual bool PostUpdate() virtual bool Draw() Text(int x, int y, UIElement* father, _TTF_Font* font, p2SString &text, bool dragable) ~Text() UIElement* GetFather() bool MouseUnderElement(int x, int y) bool Start() void Drag(int x, int y) bool Draw() virtual void Move() virtual void HandleInput() ScrollBar Window Assumption 10 SDL_Rect* thumbIdle SDL_Rect* thumbPressed const SDL_Rect* rect int value j1Module* listener Window(int x, int y, UIElement* father, SDL_Rect* rect, Text(int x, int y, UIElement* father, _TTF_Font* font, p2SString &text, bool dragable) bool dragable) ~Text() ~Window() bool Start() bool Start() bool Draw() bool Draw(