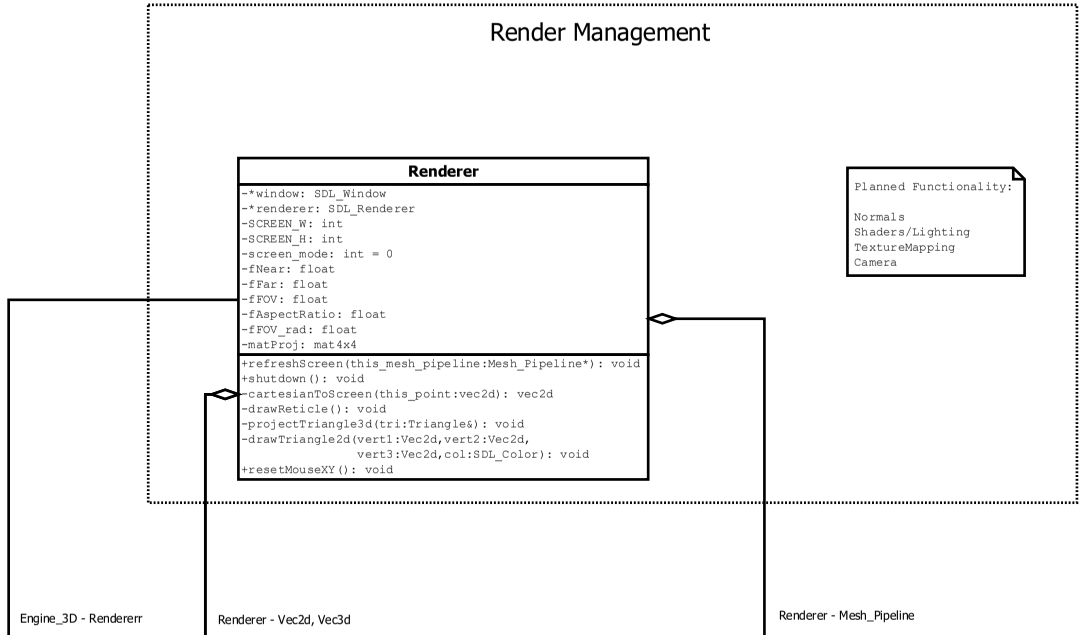
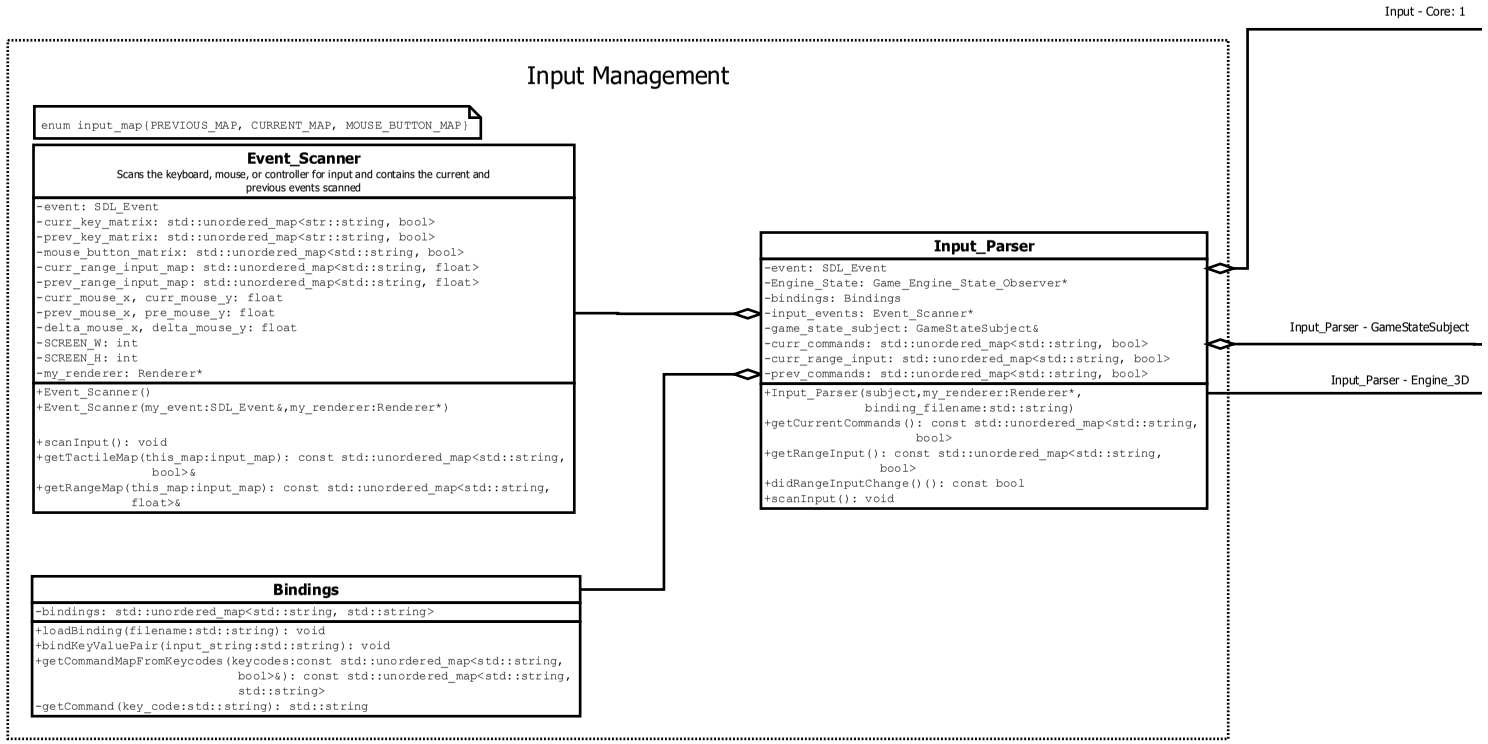
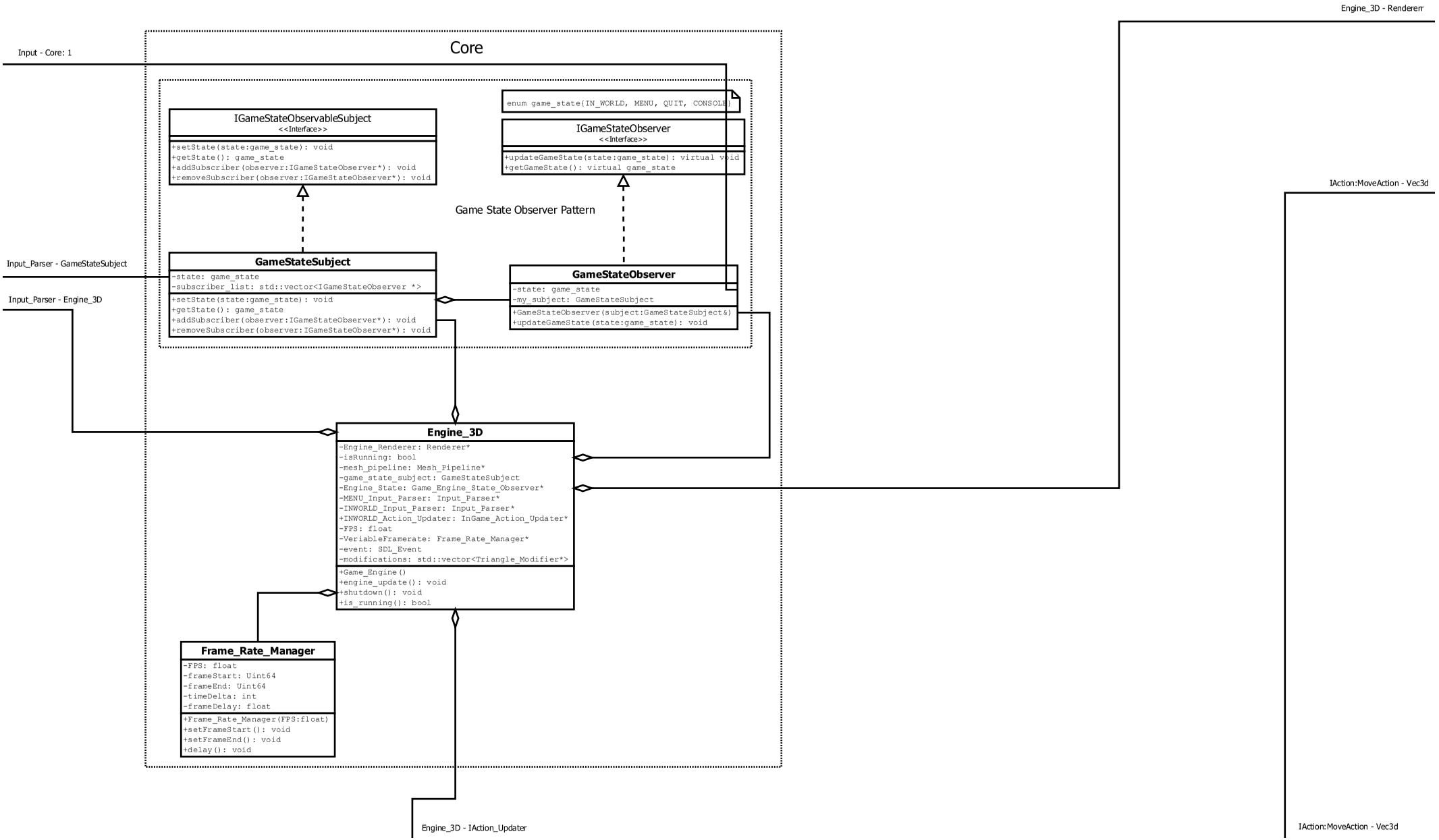
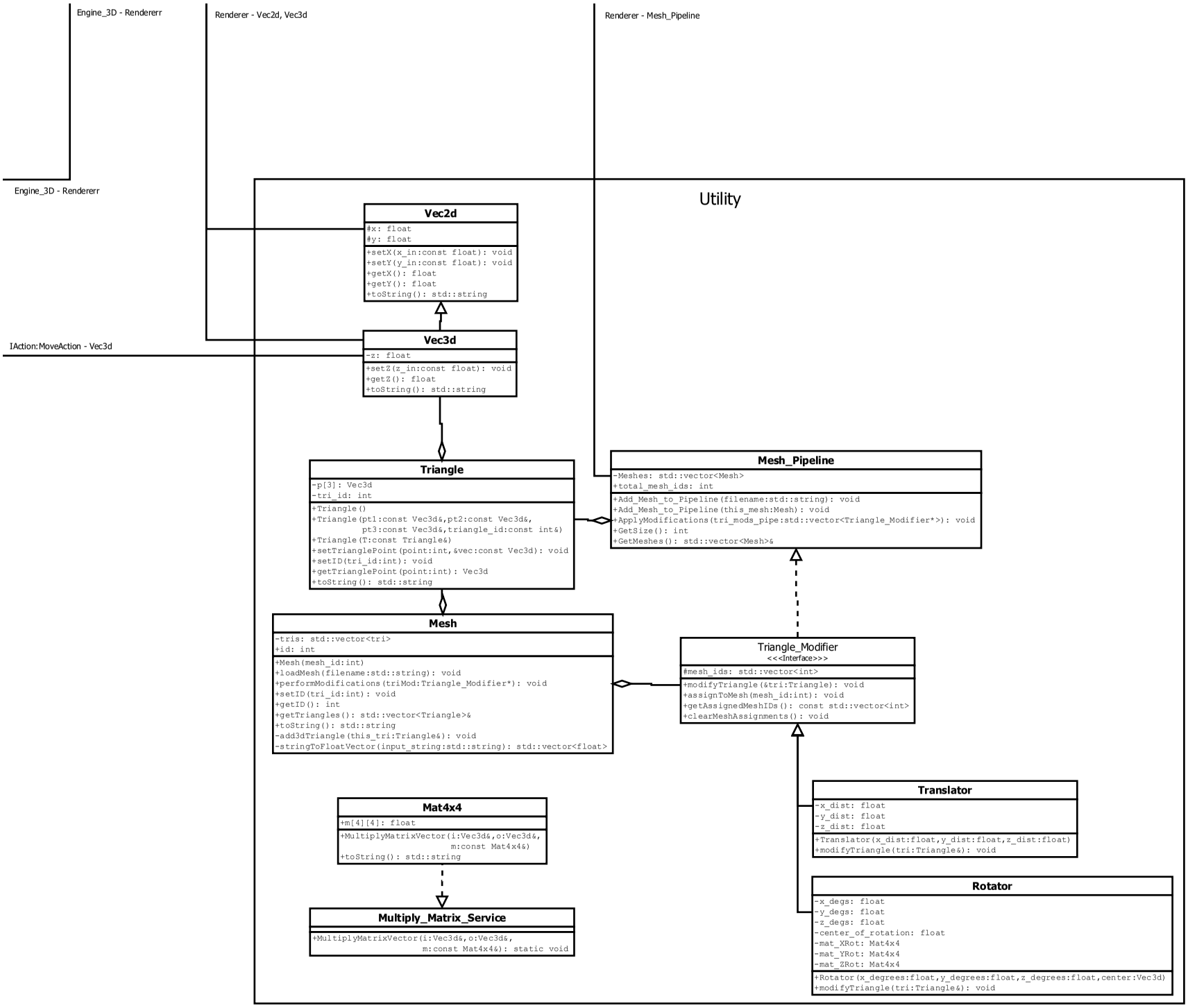
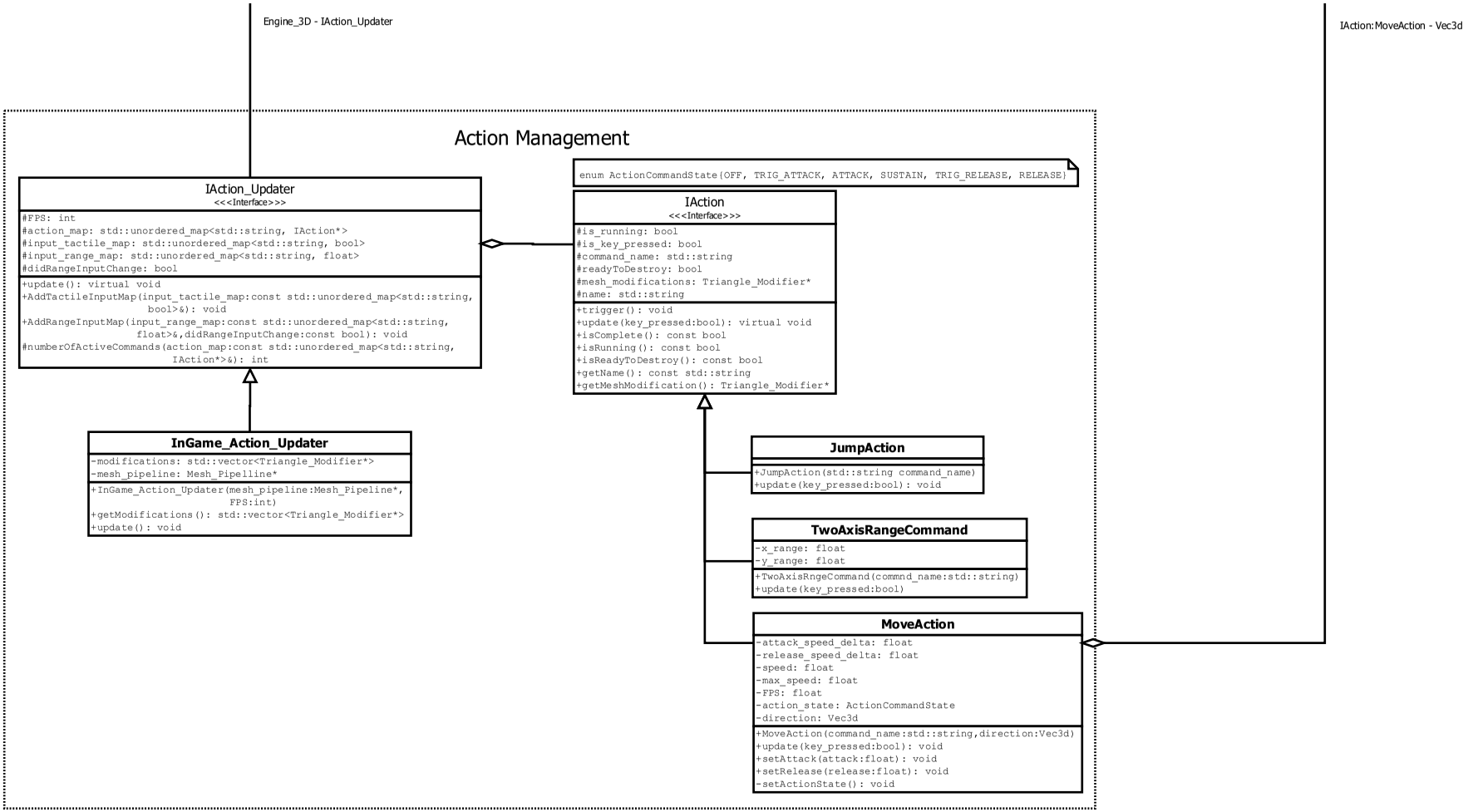

Render Management
Renderer
-*window: SDL_Window
-*renderer: SDL_Renderer
-SCREEN_W: int
-SCREEN_H: int
-screen_mode: int = 0
-fNear: float
-fFar: float
-fFOV: float
-fAspectRatio: float
-fFOV_rad: float
-matProj: mat4x4
+refreshScreen(this_mesh_pipeline:Mesh_Pipeline*): void
+shutdown(): void
-cartesianToScreen(this_point:vec2d): vec2d
-drawReticle(): void
-projectTriangle3d(tri:Triangle&): void
-drawTriangle2d(vert1:Vec2d,vert2:Vec2d,
vert3:Vec2d,col:SDL_Color): void
+resetMouseXY(): void
Planned Functionality:
Normals
Shaders/Lighting
TextureMapping
Camera
Engine_3D - Rendererr
Renderer - Vec2d, Vec3d
Renderer - Mesh_Pipeline


Input - Core: 1
Input Management
enum input_map{PREVIOUS_MAP, CURRENT_MAP, MOUSE_BUTTON_MAP}
Event_Scanner
Scans the keyboard, mouse, or controller for input and contains the current and previous events scanned
-event: SDL_Event
-curr_key_matrix: std::unordered_map<str::string, bool>
-prev_key_matrix: std::unordered_map<str::string, bool>
-mouse_button_matrix: std::unordered_map<std::string, bool>
-curr_range_input_map: std::unordered_map<std::string, float>
-prev_range_input_map: std::unordered_map<std::string, float>
-curr_mouse_x, curr_mouse_y: float
-prev_mouse_x, pre_mouse_y: float
-delta_mouse_x, delta_mouse_y: float
-SCREEN_W: int
-SCREEN_H: int
-my_renderer: Renderer*
+Event_Scanner()
+Event_Scanner(my_event:SDL_Event&,my_renderer:Renderer*)
+scanInput(): void
+getTactileMap(this_map:input_map): const std::unordered_map<std::string, bool>&
+getRangeMap(this_map:input_map): const std::unordered_map<std::string, float>&
Input_Parser
-event: SDL_Event
-Engine_State: Game_Engine_State_Observer*
-bindings: Bindings
-input_events: Event_Scanner*
-game_state_subject: GameStateSubject&
-curr_commands: std::unordered_map<std::string, bool>
-curr_range_input: std::unordered_map<std::string, bool>
-prev_commands: std::unordered_map<std::string, bool>
+Input_Parser(subject,my_renderer:Renderer*, binding_filename:std::string)
+getCurrentCommands(): const std::unordered_map<std::string, bool>
+getRangeInput(): const std::unordered_map<std::string, bool>
+didRangeInputChange()(): const bool
+scanInput(): void
Input_Parser - GameStateSubject
Input_Parser - Engine_3D
Bindings
-bindings: std::unordered_map<std::string, std::string>
+loadBinding(filename:std::string): void
+bindKeyValuePair(input_string:std::string): void
+getCommandMapFromKeycodes(keycodes:const std::unordered_map<std::string, bool>&): const std::unordered_map<std::string, std::string>
-getCommand(key_code:std::string): std::string


Engine_3D - Rendererr
Input - Core: 1
Core
enum game_state{IN_WORLD, MENU, QUIT, CONSOLE}
IGameStateObservableSubject
<<Interface>>
+setState(state:game_state): void
+getState(): game_state
+addSubscriber(observer:IGameStateObserver*): void
+removeSubscriber(observer:IGameStateObserver*): void
IGameStateObserver
<<Interface>>
+updateGameState(state:game_state): virtual void
+getGameState(): virtual game_state
IAction:MoveAction - Vec3d
Game State Observer Pattern
Input_Parser - GameStateSubject
GameStateSubject
-state: game_state
-subscriber_list: std::vector<IGameStateObserver *>
+setState(state:game_state): void
+getState(): game_state
+addSubscriber(observer:IGameStateObserver*): void
+removeSubscriber(observer:IGameStateObserver*): void
GameStateObserver
-state: game_state
-my_subject: GameStateSubject
+GameStateObserver(subject:GameStateSubject&)
+updateGameState(state:game_state): void
Input_Parser - Engine_3D
Engine_3D
-Engine_Renderer: Renderer*
-isRunning: bool
-mesh_pipeline: Mesh_Pipeline*
-game_state_subject: GameStateSubject
-Engine_State: Game_Engine_State_Observer*
-MENU_Input_Parser: Input_Parser*
-INWORLD_Input_Parser: Input_Parser*
+INWORLD_Action_Updater: InGame_Action_Updater*
-FPS: float
-VeriableFramerate: Frame_Rate_Manager*
-event: SDL_Event
-modifications: std::vector<Triangle_Modifier*>
+Game_Engine()
+engine_update(): void
+shutdown(): void
+is_running(): bool
Frame_Rate_Manager
-FPS: float
-frameStart: Uint64
-frameEnd: Uint64
-timeDelta: int
-frameDelay: float
+Frame_Rate_Manager(FPS:float)
+setFrameStart(): void
+setFrameEnd(): void
+delay(): void
Engine_3D - IAction_Updater
IAction:MoveAction - Vec3d


Engine_3D - Rendererr
Renderer - Vec2d, Vec3d
Renderer - Mesh_Pipeline
Engine_3D - Rendererr
Utility
Vec2d
#x: float
#y: float
+setX(x_in:const float): void
+setY(y_in:const float): void
+getX(): float
+getY(): float
+toString(): std::string
IAction:MoveAction - Vec3d
Vec3d
-z: float
+setZ(z_in:const float): void
+getZ(): float
+toString(): std::string
Triangle
-p[3]: Vec3d
-tri_id: int
+Triangle()
+Triangle(pt1:const Vec3d&,pt2:const Vec3d&,
pt3:const Vec3d&,triangle_id:const int&)
+Triangle(T:const Triangle&)
+setTrianglePoint(point:int,&vec:const Vec3d): void
+setID(tri_id:int): void
+getTrianglePoint(point:int): Vec3d
+toString(): std::string
Mesh_Pipeline
-Meshes: std::vector<Mesh>
+total_mesh_ids: int
+Add_Mesh_to_Pipeline(filename:std::string): void
+Add_Mesh_to_Pipeline(this_mesh:Mesh): void
+ApplyModifications(tri_mods_pipe:std::vector<Triangle_Modifier*>): void
+GetSize(): int
+GetMeshes(): std::vector<Mesh>&
Mesh
-tris: std::vector<tri>
+id: int
+Mesh(mesh_id:int)
+loadMesh(filename:std::string): void
+performModifications(triMod:Triangle_Modifier*): void
+setID(tri_id:int): void
+getID(): int
+getTriangles(): std::vector<Triangle>&
+toString(): std::string
-add3dTriangle(this_tri:Triangle&): void
-stringToFloatVector(input_string:std::string): std::vector<float>
Triangle_Modifier
<<<Interface>>>
#mesh_ids: std::vector<int>
+modifyTriangle(&tri:Triangle): void
+assignToMesh(mesh_id:int): void
+getAssignedMeshIDs(): const std::vector<int>
+clearMeshAssignments(): void
Translator
-x_dist: float
-y_dist: float
-z_dist: float
+Translator(x_dist:float,y_dist:float,z_dist:float)
+modifyTriangle(tri:Triangle&): void
Rotator
-x_degs: float
-y_degs: float
-z_degs: float
-center_of_rotation: float
-mat_XRot: Mat4x4
-mat_YRot: Mat4x4
-mat_ZRot: Mat4x4
+Rotator(x_degrees:float,y_degrees:float,z_degrees:float,center:Vec3d)
+modifyTriangle(tri:Triangle&): void
Mat4x4
+m[4][4]: float
+MultiplyMatrixVector(i:Vec3d&,o:Vec3d&,
m:const Mat4x4&)
+toString(): std::string
Multiply_Matrix_Service
+MultiplyMatrixVector(i:Vec3d&,o:Vec3d&,
m:const Mat4x4&): static void


Engine_3D - IAction_Updater
IAction:MoveAction - Vec3d
Action Management
IAction_Updater
<<<Interface>>>
#FPS: int
#action_map: std::unordered_map<std::string, IAction*>
#input_tactile_map: std::unordered_map<std::string, bool>
#input_range_map: std::unordered_map<std::string, float>
#didRangeInputChange: bool
+update(): virtual void
+AddTactileInputMap(input_tactile_map:const std::unordered_map<std::string, bool>&): void
+AddRangeInputMap(input_range_map:const std::unordered_map<std::string, float>&,didRangeInputChange:const bool): void
#numberOfActiveCommands(action_map:const std::unordered_map<std::string, IAction*>&): int
enum ActionCommandState{OFF, TRIG_ATTACK, ATTACK, SUSTAIN, TRIG_RELEASE, RELEASE}
IAction
<<<Interface>>>
#is_running: bool
#is_key_pressed: bool
#command_name: std::string
#readyToDestroy: bool
#mesh_modifications: Triangle_Modifier*
#name: std::string
+trigger(): void
+update(key_pressed:bool): virtual void
+isComplete(): const bool
+isRunning(): const bool
+isReadyToDestroy(): const bool
+getName(): const std::string
+getMeshModification(): Triangle_Modifier*
InGame_Action_Updater
-modifications: std::vector<Triangle_Modifier*>
-mesh_pipeline: Mesh_Pipelline*
+InGame_Action_Updater(mesh_pipeline:Mesh_Pipeline*, FPS:int)
+getModifications(): std::vector<Triangle_Modifier*>
+update(): void
JumpAction
+JumpAction(std::string command_name)
+update(key_pressed:bool): void
TwoAxisRangeCommand
-x_range: float
-y_range: float
+TwoAxisRngeCommand(commnd_name:std::string)
+update(key_pressed:bool)
MoveAction
-attack_speed_delta: float
-release_speed_delta: float
-speed: float
-max_speed: float
-FPS: float
-action_state: ActionCommandState
-direction: Vec3d
+MoveAction(command_name:std::string,direction:Vec3d)
+update(key_pressed:bool): void
+setAttack(attack:float): void
+setRelease(release:float): void
-setActionState(): void