D.U.N.E.

Group 11

Klas Flodin

Kaj Sandberg

Erik Nikkola

Anders Ljungqvist

Mikael Nilsson

5.5 Detailed design

5.5.1 Classes

WindowManager

Method summary:

setResolution	Change current resolution.
toggleFullscreen	Toggles fullscreen.
setTitle	Sets current window title.

Functional Requirements:
6.1.10.1 Setting video options

Name: WindowManager(int x, int y, boolean fs)
x specifies width
y specifies height
fs specifies fullscreen
Return value: N/A
Description: Creates a window with the specified values
Pre-conditions:
Validity Checks: Checks if input values is >=1 or else sets it to 1.
Post-conditions: WindowManager is initialized
Called by: Kernel
Calls: N/A

Name: setResolution(int x, int y)
x specifies width
y specifies height
Return value: boolean
Returns true if the resolution change succeeds.
Description: Changes the current in-game resolution to the new specified resolution.
Pre-conditions: WindowManager is initialised
Validity Checks: Checks if input values is >=1 or else sets it to 1.
Post-conditions: New resolution is set.
Called by: Kernel
Calls: N/A

Name: toggleFullscreen()
Return value: void
Description: Toggles fullscreen.
Pre-conditions: WindowManager is initialised
Validity Checks: None
Post-conditions: Sets fullscreen mode if previous condition was window mode.
Called by: Kernel
Calls: N/A

Name: setTitle(string name)

Return value: void
Description: Toggles fullscreen.
Pre-conditions: WindowManager is initialised
Validity Checks: None
Post-conditions: Sets fullscreen mode if previous condition was window mode.
Called by: Kernel
Calls: N/A

Log

Method summary:

writeLog	

Functional Requirements:
N/A

Name: writeLog(String log)
log is written to the log file.
Return value: void
Description: Writes a string to a pre-specified log file.
Pre-conditions: Logger is initialized and has a specified output.
Validity Checks: Target file is specified.
Post-conditions: String is written to file.
Called by: Kernel
Calls: N/A

TextureManager

Method summary:

use	

Functional Requirements:
N/A

Name: use(String name)
Name is the texture to be selected.
Return value: boolean
Returns true if the texture is found and selected.
Description: Selects the texture.
Pre-conditions: N/A
Validity Checks: Checks if the texture is loaded into memory, if not the texture manager checks for the texture on the file system and loads it if possible.
Post-conditions: Texture is selected
Called by: Building, Unit, Map, GUI
Calls: N/A

Tile

Method summary:

setTexture	
setBlocked	
isBlocked	
isSpice() setSpice()	

Functional Requirements:
6.1.3.2 Harvestable resources

Name: Tile(String texture, boolean blocked)
texture is the texture identifier
blocked is a boolean value that indicates if the tile is traversable.
Return value: N/A
Description: Initializes the tile with a texture and sets true if it's blocked
Pre-conditions: Map is initialized
Validity Checks: Checks that the texture exists.
Post-conditions: Tile is initialized.
Called by: Map
Calls: N/A

Name: setTexture(String texture)
texture is the texture identifier
Return value: boolean
Returns true if a new texture is set.
Description: Changes the current tile's texture.
Pre-conditions:
Validity Checks: Checks that the texture exists.
Post-conditions: New texture is set.
Called by: GameManager
Calls: N/A

Name: setBlocked(boolean blocked)
blocked is a boolean value that indicates if the tile is traversable.
Return value: void
Description: Sets the blocked attribute.
Pre-conditions: Map is initialized
Validity Checks: N/A
Post-conditions: New blocked attribute is set.
Called by: GameManager
Calls: N/A

Name: isBlocked()
Return value: boolean
Returns true if blocked attribute is set
Description: Returns true if blocked attribute is set

Pre-conditions: Tile is initialized.
Validity Checks: N/A
Post-conditions: N/A
Called by: GameManager, AI, Pathfinder
Calls: N/A

Name: isSpice()
Return value: boolean
Returns true if spice attribute is set
Description: Returns true if spice attribute is set
Pre-conditions: Tile is initialized.
Validity Checks: N/A
Post-conditions: N/A
Called by: Map, GameManager, Kernel
Calls: N/A

Name: setSpice(boolean spice)
spice is the desired change to spice status
Return value: void
Description: Changes spice status as desired
Pre-conditions: Tile is initialized.
Validity Checks: N/A
Post-conditions: N/A
Called by: GameManager
Calls: N/A

Map

Method summary:

generateMap	
loadMapFromFile	
getSize	
getTileMatrix getTile regenerateSpice	

Functional Requirements:
6.1.1.1 Starting a pre-made map
6.1.1.2 Starting a randomly generated map
6.1.3.2 Harvestable resources

Name: generateMap()
Return value: void
Description: Initiates a randomly generated map in
Pre-conditions: GameManager is initialized.
Validity Checks: N/A
Post-conditions: The Map object is fully initialized
Called by: Kernel
Calls: N/A

Name: loadMapFromFile(String mapName)
mapName is the name of the map file to load
Return value: void
Description: Loads the specified map file into the class, initializing it
Pre-conditions: GameManager is initialized.
Validity Checks: Validates that the mapName is a valid map file
Post-conditions: The Map object is fully initialized
Called by: Kernel
Calls: N/A

Name: getSize()
Return value: int[]
Returns the size of the map
Description: Returns the size of the map, x and y in tiles.
Pre-conditions: Map is fully initialized
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel
Calls: N/A

Name: getTileMatrix()
Return value: Tile[][]
Returns a tile matrix representing the full terrain of the map

Description: Returns an array of Tile objects containing the full map info
Pre-conditions: Map is fully initialized
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GameManager
Calls: N/A

Name: getTile(int[] position)
Return value: Tile
Returns a tile matrix representing the terrain at the given position
Description: Returns a tile matrix representing the terrain at the given position
Pre-conditions: Map is fully initialized
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel
Calls: N/A

Name: regenerateSpice()
Return value: void
Description: Regenerates spice in key areas of the map to prevent resource deadlocks
Pre-conditions: Map is fully initialized, game is in progress
Validity Checks: N/A
Post-conditions: Map is reseeded with spice tiles
Called by: GameManager
Calls: N/A

Building

Functional Requirements:
6.1.2.1 Building construction
6.1.3.1 Currency

Method summary:

No public methods

InputHandler

Functional Requirements:
6.1.2.5 Production shortcuts
6.1.7.1 Selecting a single unit or building
6.1.7.2 Selecting a group of units
6.1.7.3 Controlling units with the mouse
6.1.7.4 Controlling units with keyboard
6.1.8.1 Controlling units in combat

Method summary:

No public methods.

GameManager

Method summary:

initiateGame	
createUpdate	
createFull	
updateState	
updatePlayer	
storeCustomGameObject	
setState	
getState	

Functional Requirements:
6.1.1.1 Starting a pre-made map
6.1.1.2 Starting a randomly generated map
6.1.1.5 Pause
6.1.2.3 Primary production facilities
6.1.6.2 Designing units

Name: initiateGame(GameSettings gameSettings)
guiSettings is the identifier of all gui choices for the current game.
Return value: void
Description: Creates initial GameObject-, Player-, Map- and GameState-objects.
Pre-conditions: A GuiSettings-object must be created.
Validity Checks: Validates the preset GuiSettings.
Post-conditions: Game is initialized.
Called by: Kernel
Calls: N/A

Name: createUpdate()
Return value: Object gameUpdate
Returns a gameUpdate Object containing changes game since last update.
Description: Create the update object to be sent to all players.
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: NetworkServer
Calls: N/A

Name: createFull()
Return value: GameManager fullUpdate
Returns a full gamestate containing a complete game information.
Description: Creates complete update object to be sent to all players.
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: NetworkServer

Calls: N/A

Name: updateState(Event event)
event contains information to update the game.
Return value: void
Description: updates the game information according to the event.
Pre-conditions: An event has occured.
Validity Checks: The update is feasible.
Post-conditions: Update has been applied.
Called by: GameManager
Calls: N/A

Name: updatePlayer(Event playerEvent)
playerEvent contains information to update a player.
Return value: void
Description: updates the player information according to the event.
Pre-conditions: An event has occured.
Validity Checks: The update is feasible.
Post-conditions: The update has been applied.
Called by: GameManager
Calls: N/A

Name: storeCustomGameObject(CustomUnit cu)
cu is the information about the choices made in the GUI in form of a custom unit
Return value: void
Description: Stores custom GameObject.
Pre-conditions: A GuiSettings-object must be created.
Validity Checks: Validates the preset GuiSettings.
Post-conditions: A new custom unit-object is stored
Called by: Kernel
Calls: N/A

Name: setState(int state)
state identifies the state to be set
Return value: void
Description: Changes the current game state
Pre-conditions: A game is running
Validity Checks: N/A
Post-conditions: The game state is changed
Called by: GameManager
Calls: N/A

Name: getState()
Return value: int
returns the current game state
Description: Gets the current game state
Pre-conditions: A game is running
Validity Checks: N/A

Post-conditions: N/A
Called by: Kernel, GameManager
Calls: N/A

Unit

Method summary:

getWeapons	
getSpeed	
addPath	
getNextStep	
getMovementSound	
getArmor	

Functional Requirements:
6.1.2.2 Unit construction
6.1.2.4 Unit types
6.1.3.1 Currency
6.1.6.2 Designing units

Name: getWeapons()
Return value: Weapon
Returns a unit's Weapon
Description: Returns a Weapon Object
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GameManager
Calls: N/A

Name: getSpeed()
Return value: int
Returns an int with the units maximum speed.
Description: Returns an int with the units maximum speed.
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GameManager
Calls: N/A

Name: setPath(int[][] path)
Sets a movement path for a unit
Return value: void
Description: Sets the movement path of a unit to the supplied matrix
Pre-conditions: Unit exists
Validity Checks: N/A
Post-conditions: N/A
Called by: Pathfinder
Calls: N/A

Name: getNextStep()
Return value: int[]
returns the next movement for the unity if any
Description: Returns the next movement position of a unit if there any
Pre-conditions: Unit exists
Validity Checks: N/A
Post-conditions: N/A
Called by: GameManager
Calls: N/A

Name: getMovementSound()
Return value: String
Returns the filename of the movement sound
Description: Returns the name of the sound to be played while in movement
Pre-conditions: Unit exists
Validity Checks: N/A
Post-conditions: N/A
Called by: SFXHandler
Calls: N/A

Name: getArmor()
Return value: int
Returns an int with the units armor.
Description: Returns an int with the units armor
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GameManager
Calls: N/A

Weapon

Method summary:

getFiringSound	
getTravelSound	
getImpactSound	
getDamage	
getType	

Functional Requirements:
6.1.4.3 Upgrading research
6.1.6.2 Designing units

Name: getFiringSound()
Return value: String
Returns a string with a filename
Description: Returns the weapons firing sound affect
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: SFXHandler
Calls: N/A

Name: getTravelSound()
Return value: String
Returns a string with a filename
Description: Returns the weapons travelling sound affect
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: SFXHandler
Calls: N/A

Name: getImpactSound()
Return value: String
Returns a string with a filename
Description: Returns the weapons impact sound affect
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: SFXHandler
Calls: N/A

Name: getDamage()
Return value: int
Returns an int containing the damage value
Description: Returns the weapons damage
Pre-conditions: N/A

Validity Checks: N/A
Post-conditions: N/A
Called by: GameManager
Calls: N/A

Name: getType()
Return value: int
Returns an int containing weapon type
Description: Returns the weapons's type
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: GameManager
Calls: N/A

GameObject

Method summary:

getPosition	
setPosition	
render	
setTexture	
getHealth	
setHealth	
getDeathSound getOrientation	

Functional Requirements:
N/A

Name: getPosition()
Return value: int[]
Returns an int array with positional information
Description: Returns an int array with positional information of the GameObject
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: GameManager, AI, Kernel
Calls: N/A

Name: setPosition(int[] position)
position is an int array with the updated position
Return value: void
Description: Updates the position of the GameObject
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: GameManager, AI, Kernel
Calls: N/A

Name: render()
Return value: void
Description: Renders the current GameObject
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: Unit, Building, GUI, Map
Calls: N/A

Name: setTexture(string texture)
texture is the texture identifier
Return value: boolean

Returns true if a new texture is set.
Description: Changes the current GameObject's texture.
Pre-conditions: N/A
Validity Checks: Checks that the texture exists.
Post-conditions: New texture is set.
Called by: GameManager
Calls: N/A

Name: getHealth()
Return value: int
Returns an int with the current health
Description: Returns the GameObject's current health
Pre-conditions:
Validity Checks: Checks that the texture exists.
Post-conditions: New texture is set.
Called by: GameManager, Kernel, GUI
Calls: N/A

Name: setHealth(int health)
health is the unit's current health
Return value: void
Description: Sets the GameObject's current health
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: New health is set
Called by: GameManager
Calls: N/A

Name: getDeathSound()
Return value: String
Returns the filename of the death sound
Description: Returns the name of the sound to be played when unit dies
Pre-conditions: Unit exists
Validity Checks: N/A
Post-conditions: N/A
Called by: SFXHandler
Calls: N/A

Name: getOrientation()
Return value: float[]
Returns a float array with the current orientation
Description: Returns a float array with the current orientation
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: GameManager, AI, Kernel
Calls: N/A

Player

Method summary:

getName	
getFaction	
getColor	
getType	
getAvailableUnits	
getAvailableBuildings	
getAvailableResearch	
getPerformedResearch	
setResearch	
setBuildingConstruction	
setBuildingConstructionQueue	
setUnitConstruction	
setUnitConstructionQueue	
getAvailableModules	
getResources	
setResources	
setMainBuilding	

Functional Requirements:
6.1.2.1 Building construction
6.1.2.2 Unit construction
6.1.2.3 Primary production facilities
6.1.3.1 Currency
6.1.4.2 Research
6.1.4.2 Unlocking research
6.1.4.3 Upgrading research
6.1.5.2 Faction differences
6.1.8.3 Computer controlled opponent
6.1.8.4 Indestructible computer controlled neutral units
6.1.10.4 In-game name

Name: player(String name, int faction, int color, int type)
name specifies the name of the Player
faction identifies what faction the Player belongs to
color specifies what color the Player is
type specifies if the player is a local player, remote player or AI controlled player
Return value: N/A
Description: Creates a Player with the specified value
Pre-conditions:
Validity Checks: Checks if input values are valid.
Post-conditions: A Player object is created
Called by: Kernel
Calls: N/A

Name: getName()
Return value: String

Returns the name of the Player
Description: Returns the name of the Player
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, NetworkManager, GUI
Calls: N/A

Name: getFaction()
Return value: int
Returns the faction of the Player
Description: Returns the faction of the Player
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, NetworkManager, GUI
Calls: N/A

Name: getColor()
Return value: int
Returns the color of the Player
Description: Returns the color of the Player
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, NetworkManager, GUI
Calls: N/A

Name: getType()
Return value: int
Returns the type of the Player
Description: Returns the type of the Player
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, NetworkManager, GUI
Calls: N/A

Name: getAvailableUnits()
Return value: int[]
Returns which units the player can build
Description: Returns an integer array containing unit identifiers
Pre-conditions: Player exists.
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GUI
Calls: N/A

Name: getAvailableBuildings()
Return value: int[]
Returns which buildings the player can build
Description: Returns an integer array containing building identifiers
Pre-conditions: Player exists.
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GUI
Calls: N/A

Name: getAvailableResearch()
Return value: int[]
Returns which research the player can build
Description: Returns an integer array containing research identifiers
Pre-conditions: Player exists.
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GUI
Calls: N/A

Name: setPerformedResearch(int research)
research is used to set what research a player is performing
Return value: void
Description: Adds a researched identifier to the performed research list
Pre-conditions: Player exists.
Validity Checks: Validates that the research exists and can be performed
Post-conditions: Player's research is started
Called by: Kernel
Calls: N/A

Name: setBuildingConstruction(int building)
Sets the players current building construction
Return value: void
Description: Sets a players current construction of a building
Pre-conditions: Player exists.
Validity Checks: Validates that the building exists and construction can be started
Post-conditions: Player's construction is started
Called by: Kernel
Calls: N/A

Name: setBuildingQueue(int[] buildings)
Sets the players current building construction queue
Return value: void
Description: Sets a players current construction queue of buildings
Pre-conditions: Player exists.
Validity Checks: Validates that the building exists and construction can be started
Post-conditions: Player's construction is started
Called by: Kernel

Calls: N/A

Name: setUnitConstruction(int unit)
Sets the players current unit construction
Return value: void
Description: Sets a players current construction of a unit
Pre-conditions: Player exists.
Validity Checks: Validates that the unit exists and construction can be started
Post-conditions: Player's construction is started
Called by: Kernel
Calls: N/A

Name: setUnitQueue(int[] units)
Sets the players current unit construction queue
Return value: void
Description: Sets a players current construction queue of unit
Pre-conditions: Player exists.
Validity Checks: Validates that the unit exists and construction can be started
Post-conditions: Player's construction is started
Called by: Kernel
Calls: N/A

Name: getAvailableCustomUnitParts()
Return value: int[]
Returns what custom unit parts are available to design a custom unit
Description: Returns what custom unit parts a player can use to design a custom unit
Pre-conditions: Player exists.
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GUI
Calls: N/A

Name: getResources()
Return value: int
Returns the amount of resources player has
Description: Returns the current amount of resources the player controls
Pre-conditions: Player exists.
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GUI
Calls: N/A

Name: setResources(int resources)
Sets a player's amount of resources
Return value: void
Description: Sets the player's resource amount
Pre-conditions: Player exists.

Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GameManager
Calls: N/A

Name: setMainBuilding(int id, int type)
id is the ID of the building to set to main building
type the type of building
Return value: void
Description: Sets the player's main building of this type
Pre-conditions: Player exists
Validity Checks: Building exists
Post-conditions: N/A
Called by: Kernel
Calls: N/A

Pathfinding

Method summary:

calculatePath	

Functional Requirements:
N/A

Name: calculatePath(int[] from, int[] to, Tile[][] terrain)
from is the deparature point
to is the departure point
terrain is a representation of the tiles of the map in a matrix
Return value: int[][]
returns an integer matrix containing the calculated path
Description: Creates a new movement path matrix
Pre-conditions: Map loaded and available.
Validity Checks: Validates that the positions are valid
Post-conditions: N/A
Called by: Kernel, AI
Calls: N/A

XMLHandler

Method summary:

saveXML	
loadXML	

Functional Requirements:
6.1.2.1 Building construction
6.1.2.2 Unit construction
6.1.2.4 Unit types
6.1.5.2 Faction differences
6.1.6.2 Designing units

Name: saveXML(CustomUnit cu, String file)
cu is a custom unit object to be saved
file is the name of the file to save the Object as
Return value: void
Description: Saves a custom unit to an XML file
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: XML file created
Called by: Kernel, GameManager
Calls: N/A

Name: loadXML(String file)
file is the XML file desired to be loaded
Return value: Object
returns the XML parsed as an object
Description: Loads any supported XML data into an object, needing to be casted before use
Pre-conditions: XML data type supported
Validity Checks: Valid XML supplied
Post-conditions: Object created
Called by: Kernel, GameManager
Calls: N/A

FileHandler

Method summary:

initialize	
Read	
write	
createStream	
getFileList	
findFile	

Functional Requirements:
6.1.1.1 Starting a pre-made map
6.1.1.3 Load
6.1.1.5 Save game state
6.1.10.3 Custom soundtrack folder

Name: initialize()
Return value: void
Description: Parses the filesystem to build an internal list of each supported filetype
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: Internal Collection of supported files is created
Called by: N/A
Calls: N/A

Name: read (String file)
file is the name of the file to load
Return value: Object
Returns the read file in the proper Object
Description: Reads and parses a binary file into an Object according to file extension
Pre-conditions: FileHandler initiated
Validity Checks: File exist and type supported
Post-conditions: Object representing the file data is created
Called by: Kernel, MusicManager, TextureManager, GUI
Calls: N/A

Name: write(String file, GameManager game)
file is the name of the file to save the data to
Return value: void
Description: Saves the whole GameManager object into a file
Pre-conditions: GameManager is initialized
Validity Checks: Validates that the filename contains no illegal characters
Post-conditions: A save file of the GameManager object is created in the filesystem
Called by: Kernel
Calls: N/A

Name: createStream(String file)
file is the name of the file to create a read stream to

Return value: InputStream
Returns a inputstream to the specified file
Description: Creates an appropiate InputStream for the specified file
Pre-conditions: FileHandler initiated
Validity Checks: File identifier is correct
Post-conditions: N/A
Called by: Kernel, MusicManager, TextureManager, GUI
Calls: N/A

Name: getFileList(int type)
type is the type of the desired file list
Return value: String[]
Returns an array with the existing files
Description: Returns a list of all the files that can be loaded
Pre-conditions: FileHandler initiated
Validity Checks: File type is correct
Post-conditions: N/A
Called by: MusicManager
Calls: N/A

AI

Method summary:

nextMove	

Functional Requirements:
6.1.8.2 Defensive buildings entering combat
6.1.8.3 Computer controlled opponent
6.1.8.4 Indestructible computer controlled neutral units

Name: nextMove()
Return value: void
Description: Initiates the next move by the AI
Pre-conditions: AI Object exists and is initialized
Validity Checks: N/A
Post-conditions: GameManager objects edited according to AI move
Called by: Kernel
Calls: N/A

MusicManager

Method summary:

pause	
play	
setVolume getVolume	

Functional Requirements:
6.1.10.2 Setting audio volume
6.1.10.3 Custom soundtrack folder

Name: pause()
Return value: void
Description: Pauses the current playback
Pre-conditions: Music is played
Validity Checks: N/A
Post-conditions: Music is paused
Called by: GUI
Calls: N/A

Name: play()
Return value: void
Description: Starts playback of music
Pre-conditions: No music is played
Validity Checks: Validates that no music is played
Post-conditions: Plays music
Called by: GUI
Calls: N/A

Name: setVolume(int volume)
volume is the desired value to change volume to
Return value: void
Description: Changes volume of playback
Pre-conditions: N/A
Validity Checks: Validates that the value is within range
Post-conditions: N/A
Called by: Kernel
Calls: N/A

Name: getVolume()
Return value: int
Returns the current volume
Description: Returns the current value of the volume
Pre-conditions: MusicManager initialized
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GUI
Calls: N/A

SFXHandler

Method summary:

playSFX	
setVolume getVolume	

Functional Requirements:
6.1.10.2 Setting audio volume

Name: playSFX(String filename, int[] objectPosition, int[] cameraPosition)
filename is the name of the sound to be played
objectPosition the absolute position of the unit
cameraPosition is the absolute position of the user camera over the map
Return value: void
Description: Plays the given objects sound using the OpenAL support library
Pre-conditions: Unit exists
Validity Checks: N/A
Post-conditions: Sound played
Called by: Unit, Building, Weapon, GameManager, GUI
Calls: N/A

Name: setVolume(int volume)
volume is the desired value to change volume to
Return value: void
Description: Changes volume of playback
Pre-conditions: N/A
Validity Checks: Validates that the value is within range
Post-conditions: N/A
Called by: Kernel
Calls: N/A

Name: getVolume()
Return value: int
Returns the current volume
Description: Returns the current value of the volume
Pre-conditions: SFXHandler initialized
Validity Checks: N/A
Post-conditions: N/A
Called by: Kernel, GUI
Calls: N/A

NetworkManager

Method summary:

createConnection	
sendNetworkObject getStatus	
getUpdateState	
getFullState	

Functional Requirements:
6.1.9.1 Starting a multiplayer game
6.1.9.3 Multiplayer chat
6.1.9.4 Multiplayer cheat control

Name: createConnection(String address)
address is the IP address of the host
Return value: void
Description: Creates a connection to be used for all network traffic
Pre-conditions: NetworkServer on host is started
Validity Checks: Valid IP address
Post-conditions: Connection established with server
Called by: Kernel
Calls: N/A

Name: sendNetworkObject(Object object)
object is the object to be sent over the network
Return value: void
Description: Sends a network object to the server
Pre-conditions: NetworkServer on host is started
Validity Checks: Valid IP address
Post-conditions: Connection established with server
Called by: Kernel
Calls: N/A

Name: getStatus()
Return value: int
Description: Returns an int with current status
Pre-conditions: Game is up and running
Validity Checks: N/A
Post-conditions: Status-int returned to the calling method
Called by: Kernel
Calls: N/A

Name: getUpdateState()
Return value: Object
Description: Returns the updated state
Pre-conditions: Update is available in NetworkManager buffer
Validity Checks: N/A
Post-conditions: Status-object returned to the calling method

Called by: Kernel, GameManager
Calls: N/A

Name: getFullState()
Return value: GameManager
Description: Returns the complete updated GameManager
Pre-conditions: Update is available in NetworkManager buffer
Validity Checks: N/A
Post-conditions: Complete GameManager-object returned to the calling method
Called by: GameManager
Calls: N/A

NetworkServer

Method summary:

startListener	
stopListener	
broadcastFullState	
broadcastUpdateState	

Functional Requirements:
6.1.6.4 Multiplayer designs
6.1.9.1 Starting a multiplayer game
6.1.9.2 Request multiplayer team
6.1.9.3 Multiplayer chat
6.1.9.4 Multiplayer cheat control

Name: startListener()
Return value: void
Description: Starts a listener for incoming network connections
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: Connections can be accepted by clients
Called by: Kernel
Calls: N/A

Name: stopListener()
Return value: void
Description: Stops the listener
Pre-conditions: Listener has to be active
Validity Checks: Validates that the listener is active
Post-conditions: Connections will not be accepted by clients any longer
Called by: Kernel
Calls: N/A

Name: broadcastFullState(GameManager gm)
gm is the current gamestate
Return value: void
Description: Sends the current gamestate to all clients
Pre-conditions: Clients are connected
Validity Checks: Valid IP address and port
Post-conditions: Connection established with all clients
Called by: Kernel
Calls: N/A

Name: broadcastUpdateState(Object update)
update is the updated gamestate
Return value: void
Description: Sends the update gamestate to all clients
Pre-conditions: Clients are connected

Validity Checks: Valid IP address and port
Post-conditions: Connection established with all clients
Called by: Kernel
Calls: N/A

RenderStateManager

Method summary:

setState		
getState		

Functional Requirements:
N/A

Name: setState(Object rs)
rs is a new render state the renderer is set to.
Return value: void
Description: Sets the render state of the renderer
Pre-conditions: WindowsManager is initialized
Validity Checks: N/A
Post-conditions: Renderer's new state has been set.
Called by: GUI, GameObjects, Map
Calls: N/A

Name: getState()
Return value: Object
Object contains the current render state
Description: Returns and object with the render state
Pre-conditions: WindowManager has been initialized
Validity Checks: N/A
Post-conditions: Connections will not be accepted by clients any longer
Called by: GUI, GameObjects, Map
Calls: N/A

GUI

Method summary:

createSurface	
createButton	
Render	
listenInput	

Functional Requirements:

6.1.1.1 Starting a pre-made map
6.1.1.2 Starting a randomly generated map
6.1.1.3 Load
6.1.1.4 Save
6.1.1.5 Pause
6.1.2.1 Building construction
6.1.2.2 Unit construction
6.1.2.3 Primary production facilities
6.1.4.1 Research
6.1.5.1 Faction selection
6.1.5.2 Faction differences
6.1.6.1 Design dialogue access
6.1.6.2 Designing units
6.1.7.1 Selecting a single unit or building
6.1.7.2 Selecting a group of units
6.1.9.1 Starting a multiplayer game
6.1.10.1 Setting video options
6.1.10.2 Setting audio volume
6.1.10.4 In-game name
6.1.11.1 Quit the game

Name: createSurface(int a, int b int x, int y, float r, float g, float b, float a)
a is the screen coordinate x-value
b is the screen coordinate y-value
x is the surface's width
y is the surface's height
r is the color value (red)
g is the color value (green)
b is the color value (blue)
a is the alpha value
Return value: int
Returns a surface identifier
Description: Creates a surface window and returns an identifier to the surface
Pre-conditions: WindowsManager is initialized
Validity Checks: N/A
Post-conditions: A new surface has been created
Called by: Kernel
Calls: N/A

Name: createButton(string text, string texture)
text is a string to be written on the button
texture is a texture identifier to be shown on the button
Return value: int
Returns an identifier to the the button
Description: Create a button and returns an identifier
Pre-conditions: WindowManager has been initialized
Validity Checks: N/A
Post-conditions: A button has been created.
Called by: Kernel
Calls: N/A

Name: render()
Return value: void
Description: Renders the GUI
Pre-conditions: WindowManager has been initialized
Validity Checks: N/A
Post-conditions: The GUI is rendered.
Called by: Kernel
Calls: N/A

Name: input(Object o)
Object takes an input and interprets it and calls the appropriate function
Return value: void
Description: Takes input and calls an appropriate function
Pre-conditions: N/A
Validity Checks: N/A
Post-conditions: New function is called
Called by: Kernel
Calls:

Kernel

Functional Requirements:
All

Method summary:

No public methods.

5.5.2 Data dictionaries

Research

Field summary:	Type:
id	int
name	String
prerequisite	int[]
time	int
unlocksBuilding	int[]
unlocksUnit	int[]
unlocksCustomUnitPart	int[]
upgradesBuilding	int[][]
upgradesCustomUnitPart	int[][]
upgradesUnit	int[][]
faction	int

Functional Requirements:
6.1.4.1 Research
6.1.4.2 Unlocking research
6.1.4.3 Upgrading research
6.1.5.2 Faction differences
6.1.6.2 Designing units

Name: id
Description: Identifier for this particular research
Dependencies: none
Integrity: Must fit in a 32bit signed integer

Name: name
Description: Name of the research object
Dependencies: none
Integrity: Must fit in a String object and contain only alphanumerical characters

Name: prerequisite
Description: Previous research needed to perform this research
Dependencies: Value has to be a known research identifier
Integrity: Must fit in a 32bit signed integer

Name: time
Description: Time needed in seconds to perform this research
Dependencies: none
Integrity: Must fit in a 32bit signed integer

Name: unlocksBuilding
Description: Array of identifiers of buildings that this research unlocks
Dependencies: Must be a known building identifier

Integrity: Must fit in a 32bit signed integer

Name: unlocksUnit
Description: Array of identifiers of units that this research unlocks
Dependencies: Must be a known unit identifier
Integrity: Must fit in a 32bit signed integer

Name: unlocksCustomUnitPart
Description: Array of identifiers of custom unit parts that this research unlocks
Dependencies: Must be a known custom unit part identifier
Integrity: Must fit in a 32bit signed integer

Name: upgradesBuilding
Description: 3 x n matrix, n is the amount of upgraded buildings with this research. First value identifies the buildings that this research upgrades, second value what stat is changed, third value specifies the amount changed of the value.
Dependencies: Must be known building and stats identifiers.
Integrity: All values must fit in 32bit signed integers.

Name: upgradesUnit
Description: 3 x n matrix, n is the amount of upgraded units with this research. First value identifies the units that this research upgrades, second value what stat is changed, third value specifies the amount changed of the value.
Dependencies: Must be known units and stats identifiers.
Integrity: All values must fit in 32bit signed integers.

Name: upgradesCustomUnitPart
Description: 3 x n matrix, n is the amount of upgraded custom unit parts with this research.. First value identifies the custom unit part that this research upgrades, second value what stat is changed, third value specifies the amount changed of the value.
Dependencies: Must be known custom unit part and stats identifiers.
Integrity: All values must fit in 32bit signed integers.

Name: faction
Description: Defines for what factions this research is available
Dependencies: Must be a known faction identifier.
Integrity: All values must fit in 32bit signed integers.

Units

Field summary:	Type:
id	int
name	String
prerequisite	int[]
time	int
faction	int
type	int
cost	int
health	int
speed	int
weapon	int
armor	int

Functional Requirements:
6.1.2.2 Unit construction
6.1.2.4 Unit types
6.1.4.1 Research
6.1.5.2 Faction differences

Name: id
Description: Identifier for this particular unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer

Name: name
Description: Name of this unit
Dependencies: none
Integrity: Must fit in a String object and contain only alphanumerical characters

Name: prerequisite
Description: Specifies what research is needed to construct this unit.
Dependencies: Must be a known research identifier.
Integrity: Must fit in a 32bit signed integer.

Name: time
Description: Specifies the amount of time in seconds to construct one of this unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: faction
Description: Specifies what faction can build this unit.
Dependencies: Must be a known faction identifier.
Integrity: Must fit in a 32bit signed integer.

Name: type

Description: Specifies type of unit
Dependencies: Must be a known type identifier.
Integrity: Must fit in a 32bit signed integer.

Name: cost
Description: Specifies the cost of the unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: health
Description: Specifies the maximum health of the unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: speed
Description: Specifies the speed of the unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: weapon
Description: Specifies the weapon of the unit
Dependencies: Must be a known weapon identifier.
Integrity: Must fit in a 32bit signed integer.

Name: armor
Description: Specifies the armor of the unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Buildings

Field summary:	Type:
id	int
name	String
prerequisite	int[]
time	int
faction	int
cost	int
health	int

Functional Requirements:
6.1.2.1 Building construction
6.1.4.1 Research
6.1.5.2 Faction differences

Name: id
Description: Identifier for this particular building
Dependencies: none
Integrity: Must fit in a 32bit signed integer

Name: name
Description: Name of this building
Dependencies: none
Integrity: Must fit in a String object and contain only alphanumerical characters

Name: prerequisite
Description: Specifies what research is needed to construct this building.
Dependencies: Must be a known research identifier.
Integrity: Must fit in a 32bit signed integer.

Name: time
Description: Specifies the amount of time in seconds to construct one of this building
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: faction
Description: Specifies what faction can build this building.
Dependencies: Must be a known faction identifier.
Integrity: Must fit in a 32bit signed integer.

Name: cost
Description: Specifies the cost of the building
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: health
Description: Specifies the maximum health of the unit

Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Factions

Field summary:	Type:
id	int
name	String

Functional Requirements:
6.1.5.1 Faction selection
6.1.5.2 Faction differences

Name: id
Description: Identifier for this faction
Dependencies: none
Integrity: Must fit in a 32bit signed integer

Name: name
Description: Name of this faction
Dependencies: none
Integrity: Must fit in a String object and contain only alphanumerical characters

CustomUnitPart

Field summary:	Type:
id	int
name	String
prerequisite	int[]
cost	int
faction	int
value	int
type	int

Functional Requirements:
6.1.5.2 Faction differences
6.1.6.2 Designing units
6.1.6.3 Design budget

Name: id
Description: Identifier for this particular custom unit part
Dependencies: none
Integrity: Must fit in a 32bit signed integer

Name: name
Description: Name of this custom unit part
Dependencies: none
Integrity: Must fit in a String object and contain only alphanumerical characters

Name: prerequisite
Description: Specifies what research is needed to use this custom unit part
Dependencies: Must be a known research identifier.
Integrity: Must fit in a 32bit signed integer.

Name: cost
Description: Specifies the custom design cost this custom unit part adds
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: faction
Description: Specifies what faction can use this custom unit part.
Dependencies: Must be a known faction identifier.
Integrity: Must fit in a 32bit signed integer.

Name: value
Description: Specifies how much the stat tied to this type of custom unit part is changed
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: type

Description: Specifies what type of custom unit part
Dependencies: Must be a known custom unit part type.
Integrity: Must fit in a 32bit signed integer.

CustomUnit

Field summary:	Type:
id	int
name	String
prerequisite	int[]
cost	int
faction	int
value	int
type	int
health	int
speed	int
weapon	int
armor	int

Functional Requirements:
6.1.2.2 Unit construction
6.1.2.4 Unit types
6.1.4.1 Research
6.1.5.2 Faction differences
6.1.6.2 Designing units

Name: id
Description: Identifier for this particular custom unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer

Name: name
Description: Name of this custom unit
Dependencies: none
Integrity: Must fit in a String object and contain only alphanumerical characters

Name: prerequisite
Description: Specifies what research is needed to construct this custom unit.
Dependencies: Must be a known research identifier.
Integrity: Must fit in a 32bit signed integer.

Name: time
Description: Specifies the amount of time in seconds to construct one of this custom unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: faction
Description: Specifies what faction can build this custom unit.
Dependencies: Must be a known faction identifier.
Integrity: Must fit in a 32bit signed integer.

Name: type
Description: Specifies type of custom unit
Dependencies: Must be a known type identifier.
Integrity: Must fit in a 32bit signed integer.

Name: cost
Description: Specifies the cost of the custom unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: health
Description: Specifies the maximum health of the custom unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: speed
Description: Specifies the speed of the custom unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

Name: weapon
Description: Specifies the weapon of the custom unit
Dependencies: Must be a known weapon identifier.
Integrity: Must fit in a 32bit signed integer.

Name: armor
Description: Specifies the armor of the custom unit
Dependencies: none
Integrity: Must fit in a 32bit signed integer.

GameSettings

Field summary:	Type:
mapname	String
type	bit
players	int
playersName	String[]
playersFaction	int[]
hostAddress	String

Functional Requirements:
6.1.1.1 Starting a pre-made map
6.1.1.2 Starting a randomly generated map
6.1.5.1 Faction Selection
6.1.9.1 Starting a Multiplayer game
6.1.9.2 Request Multiplayer team
6.1.10.4 In-game name

Name: mapname
Description: Name of the map to load, *random* if a random map
Dependencies: Map name must exist as a map file or be *random*
Integrity: Must fit in a String object and contain only alphanumerical characters

Name: type
Description: Specifies type of game
Dependencies: Must be a known game type.
Integrity: Must fit in a bit.

Name: players
Description: Amount of players in the game.
Dependencies: Must be no more than 8 and no less than 2.
Integrity: Must fit in a 32bit signed integer.

Name: playersName
Description: Name of all the players in the game
Dependencies: Must be no more than 8 and no less than 2.
Integrity: Must fit in a String object and contain only alphanumerical characters

Name: playersFaction
Description: Faction of each player in the game
Dependencies: Must be known faction identifier
Integrity: Must fit in a 32bit signed integer.

Name: hostAdress
Description: IP address of the host computer.
Dependencies: Must be a valid IP address representing the host machine
Integrity: Must fit the pattern xxx.xxx.xxx.xxx, where x is a number 0-9.

5.5.3 Enumerations

Terrain	List of all terrains
Weapons	List of all weapons
Stat	List of all stats
CustomUnitPartType	List of all custom unit part types
UnitType	List of all unit types
BuildingType	List of all buildings
EventType	List of all events
Color	List of player colors
FileType	List of all file types

Functional requirements:
6.1.2.4 Unit types
6.1.3.2 Harvestable resources
6.1.4.1 Research
6.1.6.2 Designing units

5.5.4 Cross reference

6.1.1.1 Starting a pre-made map	GUI GameSettings FileHandler GameManager Map
6.1.1.2 Starting a randomly generated map	GameSettings GUI GameManager Map
6.1.1.3 Load	GUI FileHandler
6.1.1.4 Save	GUI
6.1.1.5 Pause	GUI GameManager
6.1.1.5 Save game state	FileHandler
6.1.10.2 Setting audio volume	GUI SFXHandler MusicManager WindowManager
6.1.10.3 Custom soundtrack folder	MusicManager FileHandler
6.1.10.4 In-game name	GUI GameSettings Player
6.1.11.1 Quit the game	GUI
6.1.2.1 Building construction	GUI Buildings XMLHandler Player Building
6.1.2.2 Unit construction	GUI CustomUnit Units XMLHandler Player Unit
6.1.2.3 Primary production facilities	GUI Player GameManager
6.1.2.4 Unit types	Enumerations CustomUnit Units XMLHandler Unit
6.1.2.5 Production shortcuts	InputHandler
6.1.3.1 Currency	Player Unit

6.1.3.2 Harvestable resources	Building Enumerations Map Tile
6.1.4.2 Research	Player GUI Enumerations CustomUnit Buildings Units Research
6.1.4.2 Unlocking research	Research Player
6.1.4.3 Upgrading research	Weapon Research Player
6.1.5.1 Faction selection	GUI GameSettings Factions
6.1.5.2 Faction differences	GUI CustomUnit CustomUnitPart Factions Buildings Units
6.1.6.1 Design dialogue access	GUI
6.1.6.2 Designing units	GUI Enumerations CustomUnit CustomUnitPart Research XMLHandler Weapon
6.1.6.3 Design budget	CustomUnitPart
6.1.6.4 Multiplayer designs	NetworkServer
6.1.7.1 Selecting a single unit or building	GUI InputHandler
6.1.7.2 Selecting a group of units	GUI InputHandler
6.1.7.3 Controlling units with the mouse	InputHandler
6.1.7.4 Controlling units with keyboard	InputHandler
6.1.8.1 Controlling units in combat	InputHandler
6.1.8.2 Defensive buildings entering combat	AI
6.1.8.3 Computer controlled opponent	AI Player
6.1.8.4 Indestructible computer controlled neutral units	AI Player
6.1.9.1 Starting a Multiplayer game	GameSettings

	GUI NetworkServer NetworkManager
6.1.9.2 Request Multiplayer team	GameSettings NetworkServer
6.1.9.3 Multiplayer chat	NetworkServer NetworkManager
6.1.9.4 Multiplayer cheat control	NetworkServer NetworkManager

5.6 Package Diagram