Project Hellknöw

Group 3
Henrik Sandström
Jonas Lindmark
Carl-Fredrik Sundlöf
Tim Hao Li

Table of Contents

5.5 Detailed Design

Game Engine

Objects and Data Structures:

Object Name: physics

Type: Physics

Description

Will be an object of physics that the game engine uses to update positions and other status of all objects in the game

Funcional Requirements Reference

4.1.1 Objects, 4.1.2 Obstacles, 4.1.3 Weapons, 4.1.4 Players

Object Name: objects

Type: LinkedList<Object> (from java.util)

Description

Will be an LinkedList with all the objects in the game, both stationary and movable.

Funcional Requirements Reference

4.1.1 Objects, 4.1.2 Obstacles, 4.1.3 Weapons, 4.1.4 Players

Object Name: comm

Type: Communication

Description

Will be an object of communication that the game engine uses to establish a connection and then also us it.

Funcional Requirements Reference

4.1.5 Network

Object Name: display

Type: Display

Description

Will be an object of display that the game engine uses to create graphical output.

Funcional Requirements Reference

4.1.1 Objects, 4.1.2 Obstacles, 4.1.3 Weapons, 4.1.4 Players

Object Name: sound

Type: Sound

Description

Will be an object of sound that the game engine uses to create audible output.

Object Name: input

Type: Input

Description

Will be an object of input that the game engine uses to parse input arguments.

Methods:

Method Name: void start()

Takes no parameters.

Return Value: none

Description

Initiates the first screen that the game is going to display. This is going to be the Main Menu.

Data structure used: None

Pre-conditions: This functions assumes that nothing has yet been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: The main menu is displayed and run() is called.

Called by: Game Engine

Calls: run(), showMainMenu().

Method Name: void run()

Takes no parameters.

Return Value: none

Description

Contains the main game loop from which the game is handled. The game is over when this method is terminated.

Data structure used: None

Pre-conditions: Assumes that the MainMenu is currently displayed to the user.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: The game has exited and is not in the computers memory anymore.

Called by: start().

Calls: Should be able to call any method.

Method Name: void hostGame(int Players)

Players is the number of players that will play the game. Players will be either 1 or 2.

Return Value: none

Description

This method will tell communication to establish a socket for the server(s) to communicate on.

Data structure used: None

Pre-conditions: A game is not running.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: The game is either started if the connection is established or the main menu is displayed again.

Called by: run().

Calls: hostConnection(), showMainMenu().

Method Name: void joinGame(String IP)

IP is a string with the IP of the hosting server. IP will be obtained from the user through a form.

Return Value: none

Description

This method will establish a connection to the hosting user.

Data structure used: None

Pre-conditions: Assumes that the user is connected to a network.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: The players are either connected or the main menu is displayed again.

Called by: run().

Calls: establishConnection(IP), showMainMenu().

Method Name: void performAction(String action)

Action is the name of the action to perform.

Return Value: none

Description

Gets information from display and performs a certain action

Data structure used: None

Pre-conditions: None

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: The action is performed.

Called by: mouseClicked().

Calls: parse

Communication

Objects and Data Structures:

Object Name: gePointer

Type: Game Engine

Description

A reference back to game engine.

Methods:

Method Name: socket establishConnection(String IP)

IP is a string with the IP of the hosting server. IP will be obtained from the user with a form.

Return Value: Socket which is established, or null if a connection could not be established.

Description

This method will connect the user to the hosting user.

Data structure used: None

Pre-conditions: Assumes that it will be possible to establish the connection.

Validity Checks, Errors, and other Anomalous Situations: Checks that the connection can be established.

Post-conditions: The game is either started if the connection is established or the main menu is displayed again.

Called by: joinGame().

Calls: Uses Java sockets to create the connection.

Funcional Requirements Reference

4.1.5 Network

Method Name: socket hostConnection()

Takes no parameters.

Return Value: Socket which is established, or null if a connection could not be established.

Description

This method will create the socket on which to communicate.

Data structure used: None

Pre-conditions: Assumes that it will be possible to establish the connection.

Validity Checks, Errors, and other Anomalous Situations: Checks that the connection can be established.

Post-conditions: The game is either started if the connection is established or the main menu is displayed again.

Called by: joinGame().

Calls: Uses java sockets to create the connection.

Funcional Requirements Reference

4.1.5 Network

Method Name: void send(String data)

data is the information that is to be sent to the other server.

Return Value: none.

Description

Send sends information over the socket.

Data structure used: None

Pre-conditions: Assumes that there is a established connection

Validity Checks, Errors, and other Anomalous Situations: Checks that the connection is established.

Post-conditions: The information has been sent.

Called by: Game engine.

Calls: Uses java sockets to send information.

Method Name: String receive()

Takes no parameters.

Return Value: String with the instruction received.

Description

Receive receives information over the socket.

Data structure used: None

Pre-conditions: Assumes that there is a established connection

Validity Checks, Errors, and other Anomalous Situations: Checks that the connection is established.

Post-conditions: The information has been received.

Called by: Game engine.

Calls: Uses java sockets to receive information.

Input

Objects and Data Structures:

Object Name: gePointer

Type: Game Engine

Description

A reference back to game engine.

Methods:

Method Name: void performEvent(keyEvent input)

Input is the keyEvent to perform.

Return Value: none

Description

Preforms the action bind to the keyEvent specified in input.

Data structure used: None

Pre-conditions: A game is running.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: The game has exited and is not in the computers memory anymore.

Called by: Display.

Calls: None

Sounds

Objects and Data Structures:

Object Name: gePointer

Type: Game Engine

Description

A reference back to game engine.

Funcional Requirements Reference

Methods:

Method Name: wav getSound(String name)

name: The name of the sound.

Return Value: wav sound

sound: The wav mapped to name, if name does not exist, return null.

Description

The function takes a name and searches it's database for a sound with a key identical to name. It then returns that Sound object.

Data structure used: A hashmap with value type Sound and key type String.

Pre-conditions: The database of Sounds is non empty.

Validity Checks, Errors, and other Anomalous Situations: If the inputted name doesn't exist in the database, returns null.

Post-conditions: None.

Called by: The Sound class will call this method when it wants to play a sound associated with an event.

Calls: None.

Method Name: void loadSounds()

Return Value: Nothing.

Description: When this method is invoked, it will attempt to load all sounds with their respective keys from our file structure into our internal hashmap.

Data structure used: A hashmap with value type Sound and key type String.

Pre-conditions: The files needed are located in the correct folder and are of non-zero size.

Validity Checks, Errors, and other Anomalous Situations: If any file that is needed cannot be found or otherwise cannot be opened the method will throw FileNotFoundException.

Post-condition: The internal database will be filled with the sounds described in the filesystem.

Called by: It's own internal constructor.

Calls: None.

Funcional Requirements Reference

Sound

Objects and Data Structures:

Object Name: gePointer

Type: Game Engine

Description

A reference back to game engine.

Funcional Requirements Reference

Object Name: soundOn

Type: boolean

Description:

If this is false no sound will be played, if true it does not affect anything.

Methods:

Method Name: void playSound(String id)

id: The id of the sound that is to be played.

Return Value: Nothing.

Description: Makes a system call to play the sound associated with the id.

Data structure used: None.

Pre-conditions: None.

Validity Checks, Errors, and other Anomalous Situations: If there is no sound associated with the id the method will not play any sound.

Post-condition: None.

Called by: Game Engine.

Calls: getSound(id), system call to play sounds.

Funcional Requirements Reference

Method Name: void setSound(boolean on)

on: The new value of soundOn.

Return Value: Nothing.

Description: Sets whether sound should be played or not.

Data structure used: None.

Pre-conditions: None.

Validity Checks, Errors, and other Anomalous Situations: None.

Post-condition: None.

Called by: Game Engine.

Calls: None.

Images

Objects and Data Structures:

Object Name: gePointer

Type: Game Engine

Description

A reference back to game engine.

Funcional Requirements Reference

Methods:

Method Name: Image getImage(String id)

id: The id of the image.

Return Value: Image img

img: The image mapped to id.

Description

The function will search its database for an image mapped to the key id and return it.

Data structure used: A hashmap with value type Image and key type String.

Pre-conditions: The database of images is non empty.

Validity Checks, Errors, and other Anomalous Situations: If the inputted name doesn't exist in the database, returns null.

Post-conditions: None.

Called by: Display.

Calls: None.

Method Name: void loadImages()

Return Value: Nothing.

Description: When this method is invoked, it will attempt to load all images with their respective keys from our file structure into our internal hashmap.

Data structure used: A hashmap with value type Sound and key type String.

Pre-conditions: The files needed are located in the correct folder and are of non-zero size.

Validity Checks, Errors, and other Anomalous Situations: If any file that is needed cannot be found or otherwise cannot be opened the method will throw FileNotFoundException.

Post-condition: The internal database will be filled with the sounds described in the filesystem.

Called by: It's own internal constructor.

Calls: None.

Graphics

This will include all methods described by the graphics engine we choose. All methods will be called by the Display class.

Display

Objects and Data Structures:

Object Name: gePointer

Type: Game Engine

Description

A reference back to game engine.

Object Name: sensitiveAreas

Type: LinkedList

Description:

Will be used to store the currently active sensitive areas.

Methods:

Method Name: void showHints()

Return Value: Nothing.

Description: This method will print the hints menu on the display.

Data structure used: None.

Pre-conditions: The display has been initiated. A game instance is not active.

Validity Checks, Errors, and other Anomalous Situations: If any of the required images are not found it will throw NullPointerException.

Post-condition: The hints field is displayed.

Called by: Game Engine.

Calls: Various methods in Graphics, getImage from Images.

Method Name: void showMainMenu()

Return Value: Nothing.

Description: This will show the Main Menu and change the mouse sensitive areas in the display correctly.

Data structure used: None.

Pre-conditions: A game instance is not active.

Validity Checks, Errors, and other Anomalous Situations: If any of the required images are not found it will throw NullPointerException.

Post-condition: The Main Menu is displayed.

Called by: Game Engine.

Calls: Various methods in Graphics, getImage from Images.

Method Name: void showIP()

Return Value: Nothing.

Description: This will show a box with the user's IP.

Data structure used: None.

Pre-conditions: A game instance is not active. The user is connected to the Internet.

Validity Checks, Errors, and other Anomalous Situations: If the IP cannot be found or the user doesn not have a connection to the Internet nothing will be displayed.

Post-condition: A box with the IP is displayed.

Called by: Game Engine.

Calls: Uses the Java Net library to find the user's IP.

Method Name: void showMultiplayerMenu()

Return Value: Nothing.

Description: This will show the Multiplayer Menu and change the mouse sensitive areas in the

display correctly.

Data structure used: None.

Pre-conditions: A game instance is not active.

Validity Checks, Errors, and other Anomalous Situations: If any of the required images are not found it will throw NullPointerException.

Post-condition: The Mulyiplayer Menu is displayed.

Called by: Game Engine.

Calls: Various methods in Graphics, getImage from Images.

Method Name: void mouseClicked(MouseEvent e)

e: The event generated when the mouse has been clicked within the display.

Return Value: Nothing.

Description: This method is automatically invoked when the mouse has been clicked. The purpose of this method is to identify what action to take depending on where the mouse was when it clicked, and calls that method. This is done by iterating through a list of sensitive areas and checking if the mouse was within that area, if found it will send that action to the Game Engine to be handled.

Data structure used: LinkedList.

Pre-conditions: The display has been initialized.

Validity Checks, Errors, and other Anomalous Situations: None.

Post-condition: The correct action is sent to Game Engine.

Called by: Java MouseListener

Calls: performAction in Game Engine.

Method Name: void keyPressed(KeyEvent e)

e: The event generated when a key has been pressed.

Return Value: Nothing.

Description: This method is automatically invoked when a key has been pressed. The method will send this KeyEvent to Input so that Input can interpret and handle the action.

Data structure used: None.

Pre-conditions: The display has been initialized. The Input pointer ip is not null.

Validity Checks, Errors, and other Anomalous Situations: None.

Post-condition: The KeyEvent has been sent to Input.

Called by: Java KeyListener.

Calls: performEvent in Input.

Object

Objects and Data Structures:

Object Name: gePointer

Type: Game Engine

Description

A reference back to game engine.

Object Name: x

Type: int

Description

The objects x coordinate

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.3 Frictional force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.1 Move horizontally, 4.1.4.2 Jump, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Object Name: y

Type: int

Description

The objects y coordinate

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.4 Gravitational Force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.2 Jump, 4.1.4.4 Climb ladders, 4.1.4.3 Crouch, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Object Name: key

Type: String

Description

The key of the sprite picture in the sprite hashmap that represent the object.

Methods:

Method name: getX ()

Return Value: int x

Returns the objects x coordinate.

Description

To know the objects x coordinate you may call the specified objects getX method to receive the x coordinate.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.3 Frictional force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.1 Move horizontally, 4.1.4.2 Jump, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Method name: setX (int x)

The input value is the new x coordinate.

Return Value: None

Description

To set the objects x coordinate you may call the specified objects setX method to set the x coordinate.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: Changes the objects x coordinate.

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.3 Frictional force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.1 Move horizontally, 4.1.4.2 Jump, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Method name: getY ()

Return Value: int y

Returns the objects y coordinate.

Description

To know the objects y coordinate you may call the specified objects getY method to receive the y coordinate.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.4 Gravitational Force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.2 Jump, 4.1.4.4 Climb ladders, 4.1.4.3 Crouch, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Method name: setY (int y)

The input value is the new y coordinate.

Return Value: None

Description

To set the objects y coordinate you may call the specified objects setY method to set the y coordinate.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: Changes the objects y coordinate.

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.4 Gravitational Force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.2 Jump, 4.1.4.4 Climb ladders, 4.1.4.3 Crouch, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Method name: getSpriteKey()

Return Value: String key

Returns the key of the sprite picture in the sprite hashmap that represent the object.

Description

When getting the sprite that represent the object you may call getSpriteKey for the specified object so that you can fetch the correct sprite from the sprite hashmap.

Data structure used: None

Pre-conditions: Assumes that the object has been created and that the sprite hashmap has beeen initialized.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: Display

Calls: None

Movable Object

Objects and Data Structures:

Object Name: vx

Type: int

Description

The objects velocity in x direction

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.3 Frictional force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.1 Move horizontally, 4.1.4.2 Jump, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Object Name: vy

Type: int

Description

The objects velocity in y direction

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.4 Gravitational Force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.2 Jump, 4.1.4.4 Climb ladders, 4.1.4.3 Crouch, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Object Name: hp

Type: int

Description

The amount of health points that the object has.

Functional Requirements Reference

4.1.4 Player: 4.1.4.9 Lose health points, 4.1.4.9 Losing health points when hit by a weapon

Object Name: dead

Type: boolean

Description

True if the object is dead otherwise false

Functional Requirements Reference

4.1.4 Player: 4.1.4.9 Lose health points, 4.1.4.9 Losing health points when hit by a weapon

Methods:

Method name: getVX ()

Return Value: int vx

Returns the objects velocity in x direction

Description

To know the objects velocity in x direction you may call the specified objects getVX method to receive the objects velocity in x direction.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.3 Frictional force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.1 Move horizontally, 4.1.4.2 Jump, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Method name: setVX (int vx)

The input value is the new velocity in x direction

Return Value: None

Description

To set the objects velocity in x direction you may call the specified objects setVX method to set the objects velocity in x direction.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: Changes the objects velocity in x direction.

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.3 Frictional force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.4 Players: 4.1.4.1 Move horizontally, 4.1.4.2 Jump, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Method name: getVY ()

Return Value: int vy

Returns the objects velocity in y direction

Description

To know the objects velocity in y direction you may call the specified objects getVY method to

receive the objects velocity in y direction.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.4 Gravitational Force

4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions

4.1.3 Weapons: 4.1.3.3 Gravity affect bullets

4.1.4 Players: 4.1.4.2 Jump, 4.1.4.4 Climb ladders, 4.1.4.3 Crouch, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Method name: setVY (int vy)

The input value is the new velocity in y direction

Return Value: None

Description

To set the objects velocity in y direction you may call the specified objects setVY method to set the objects velocity in y direction.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: Changes the objects velocity in y direction.

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.1 Objects: 4.1.1.1 Changes in movement, 4.1.1.2 Interaction, 4.1.1.4 Gravitational Force
4.1.2 Obstacles: 4.1.2.1 There will be obstacles in the game preventing the player to move in certain directions
4.13 Weapons: 4.1.3.3 Gravity affect bullets
4.1.4 Players: 4.1.4.2 Jump, 4.1.4.4 Climb ladders, 4.1.4.3 Crouch, 4.1.4.5 Pass through objects, 4.1.4.6 Changing line of sight

Method name: getHP()

Return Value: int hp
Returns the objects health points
Description
To know the objects health points you may call the specified objects getHP method to receive the objects health points.
Data structure used: None
Pre-conditions: Assumes that the object has been created.
Validity Checks, Errors, and other Anomalous Situations: None
Post-conditions: None
Called by: Physics
Calls: None
Functional Requirements Reference
4.1.4 Player: 4.1.4.9 Lose health points, 4.1.4.9 Losing health points when hit by a weapon

Method name: reciveDamage (int damage)

The input value is the damage that the object recives
Return Value: None
Description

The damage is subtracted from the objects health points. If the health points reaches below zero the boolean dead changes from false to true.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: Changes the objects velocity in y direction.

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.4 Player: 4.1.4.9 Lose health points, 4.1.4.9 Losing health points when hit by a weapon

Method name: isDead()

Return Value: boolean dead

Description

Returns the boolean dead which tells whether or not the object is alive or if it has been destroyed and therefore is dead

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: Physics

Calls: None

Functional Requirements Reference

4.1.4 Player: 4.1.4.9 Lose health points, 4.1.4.9 Losing health points when hit by a weapon

Player

Objects and Data Structures:

Object Name: weaponC

Type: Weapon

Description

The close combat weapon.

Functional Requirements Reference

4.1.3 Weapon:

Object Name: weaponR

Type: Weapon

Description

The ranged weapon.

Functional Requirements Reference

4.1.3 Weapons: 4.1.3.1 Assigned weapons, 4.1.3.2 Cool down, 4.1.3.3 Gravity affect bullets

Methods:

Method name: addWeaponC()

Return Value: None

Description

Adds the close combat weapon to the players weapon arsenal.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: GameEngine

Calls: None

Functional Requirements Reference

4.1.3 Weapons: 4.1.3.1 Assigned weapons, 4.1.3.2 Cool down, 4.1.3.3 Gravity affect bullets

Method name: addWeaponR()

Return Value: None

Description

Adds the ranged weapon to the players weapon arsenal. And sets it to the equipedWeapon.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: GameEngine

Calls: None

Functional Requirements Reference

4.1.3 Weapons: 4.1.3.1 Assigned weapons, 4.1.3.2 Cool down, 4.1.3.3 Gravity affect bullets

Method name: currentWeapon ()

Return Value: String equipedWeapon

Returns the string with the name of the currently equipped weapon.

Description

To know the weapon that the player currently has equipped you may call this function to receive the information.

Data structure used: None

Pre-conditions: Assumes that the object has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: Game Engine and the changeWeapon method

Calls: None

Functional Requirements Reference

4.1.3 Weapons: 4.1.3.1 Assigned weapons, 4.1.3.2 Cool down, 4.1.3.3 Gravity affect bullets

Method name: changeWeapon ()

Return Value: None

Description

Calls currentWeapon to see what weapon the player currently has equipped and changes the equipedWeapon to the other weapon the player has.

Data structure used: None

Pre-conditions: Assumes that the object has been created and that the player has it's two weapons initialized.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: Changes the players weapon.

Called by: GameEninge

Calls: None

Functional Requirements Reference

4.1.3 Weapons: 4.1.3.1 Assigned weapons, 4.1.3.2 Cool down, 4.1.3.3 Gravity affect bullets

Weapon

Objects and Data Structures:

Object Name: Name

Type: String

Description

The name of the weapon

Object Name: damage

Type: int

Description

The amount of damage that the weapon does

Functional Requirements Reference

4.1.3 Weapons: 4.1.3.1 Assigned weapons

Object Name: range

Type: int

Description

The range of the weapon

Functional Requirements Reference

4.1.3 Weapons: 4.1.3.1 Assigned weapons, 4.1.3.3 Gravity affect bullets

Methods:

Method name: getName()

Return Value: String name

Description

Returns the name of the weapon

Data structure used: None

Pre-conditions: Assumes that the weapon has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: GameEngine

Calls: None

Method name: getDamage()

Return Value: int damage

Description

Returns the damage of the weapon

Data structure used: None

Pre-conditions: Assumes that the weapon has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: GameEngine

Calls: None

Functional Requirements Reference

4.1.3 Weapons: 4.1.3.1 Assigned weapons, 4.1.3.3 Gravity affect bullets

Method name: getRange()

Return Value: int range

Description

Returns the range of the weapon

Data structure used: None

Pre-conditions: Assumes that the weapon has been created.

Validity Checks, Errors, and other Anomalous Situations: None

Post-conditions: None

Called by: GameEngine

Calls: None

Functional Requirements Reference

4.1.3 Weapons: 4.1.3.1 Assigned weapons, 4.1.3.3 Gravity affect bullets

5.6 Package Diagram