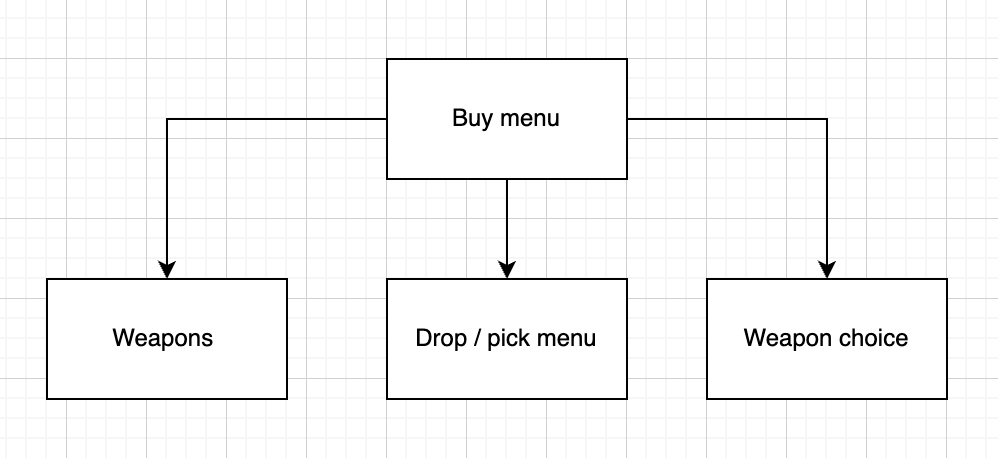
# Design section

Main menus of my program:

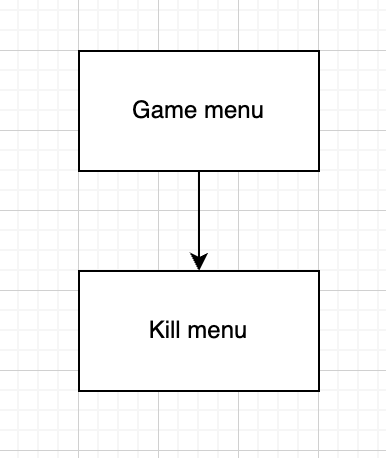
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Side choice menu | Main menu | Buy menu | Game menu | Results menu |

Submenus of each menu:

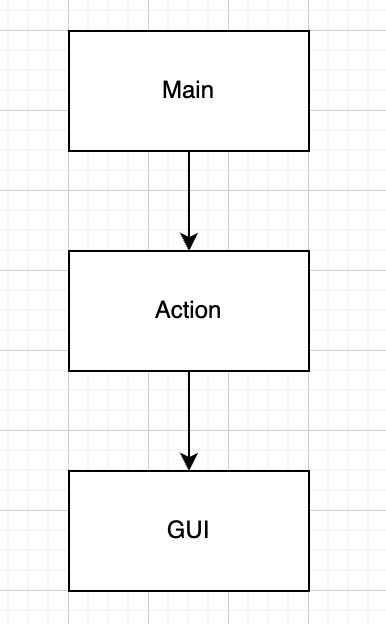
Buy menu:



Game menu



# Classes



Functions of:

Main:

* The personalized user interface
* Contains the objects
* Creates all buttons

Action:

* Allow user to press the buttons
* Allow user to enter the input
* Allows to see the indicators, such as balance and round count
* Carries out the balance calculations when the weapon is bought
* Updates all the data, such as weapons used, balance, etc.
* Create a connection between MySQL and BlueJ

GUI:

* Creates the graphical user interface

# MySQL database:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Field | Schema | Table | Type |
| 1 | IdNumCS **(primary key)** | CSGOSchema | IACompCSGO | INT |
| 2 | RoundCount | CSGOSchema | IACompCSGO | INT |
| 3 | Balance | CSGOSchema | IACompCSGO | INT |
| 4 | RoundT | CSGOSchema | IACompCSGO | INT |
| 5 | RoundCT | CSGOSchema | IACompCSGO | INT |
| 6 | Weapon1 | CSGOSchema | IACompCSGO | VARCHAR |
| 7 | Weapon2 | CSGOSchema | IACompCSGO | VARCHAR |
| 8 | Armor | CSGOSchema | IACompCSGO | VARCHAR |
| 9 | Grenade1 | CSGOSchema | IACompCSGO | VARCHAR |
| 10 | Grenade2 | CSGOSchema | IACompCSGO | VARCHAR |
| 11 | Grenade3 | CSGOSchema | IACompCSGO | VARCHAR |
| 12 | Grenade4 | CSGOSchema | IACompCSGO | VARCHAR |
| 13 | Zeus | CSGOSchema | IACompCSGO | VARCHAR |
| 14 | MainGameCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 15 | WeaponChoiceCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 16 | TCZ75orTec9 | CSGOSchema | IACompCSGO | VARCHAR |
| 17 | TDeagleORr8 | CSGOSchema | IACompCSGO | VARCHAR |
| 18 | Tmp5SDorMP7 | CSGOSchema | IACompCSGO | VARCHAR |
| 19 | CTuspsORp2000 | CSGOSchema | IACompCSGO | VARCHAR |
| 20 | CTDeagleORr8 | CSGOSchema | IACompCSGO | VARCHAR |
| 21 | CTCZ75or57 | CSGOSchema | IACompCSGO | VARCHAR |
| 22 | CTmp5SDorMP7 | CSGOSchema | IACompCSGO | VARCHAR |
| 23 | CTM4a4orM4a1s | CSGOSchema | IACompCSGO | VARCHAR |
| 24 | SideChoice | CSGOSchema | IACompCSGO | VARCHAR |
| 25 | PistolsCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 26 | MolotovBuyCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 27 | DecoyBuyCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 28 | FlashBuyCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 29 | SmokeBuyCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 30 | HEBuyCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 31 | LimitGrenadeCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 32 | PistolNullCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 33 | KillNumber | CSGOSchema | IACompCSGO | INT |
| 34 | KillEntryNumber | CSGOSchema | IACompCSGO | INT |
| 35 | HMKcondition | CSGOSchema | IACompCSGO | VARCHAR |
| 36 | BombPlant | CSGOSchema | IACompCSGO | VARCHAR |
| 37 | buttonBombPlant | CSGOSchema | IACompCSGO | VARCHAR |
| 38 | buttonSuicideReward | CSGOSchema | IACompCSGO | VARCHAR |
| 39 | buttonGrenadeUse | CSGOSchema | IACompCSGO | VARCHAR |
| 40 | buttonTeamKill | CSGOSchema | IACompCSGO | VARCHAR |
| 41 | buttonSurvive | CSGOSchema | IACompCSGO | VARCHAR |
| 42 | loseBonusCount | CSGOSchema | IACompCSGO | INT |
| 43 | winAfterPistolRound | CSGOSchema | IACompCSGO | VARCHAR |
| 44 | howDidYouWinTheRound | CSGOSchema | IACompCSGO | VARCHAR |
| 45 | incendiaryBuyCondition | CSGOSchema | IACompCSGO | VARCHAR |
| 46 | kits | CSGOSchema | IACompCSGO | VARCHAR |

` `,

Example data value:

|  |  |
| --- | --- |
| IdNumCS | 1 |
| RoundCount | 8 |
| Balance | 800 |
| RoundT | 3 |
| RoundCT | 15 |
| Weapon1 | Glock-18 |
| Weapon2 | AK-47 |
| Armor | Kevlar Vest |
| Grenade1 | Incendiary |
| Grenade2 | HE Grenade |
| Grenade3 | Decoy |
| Grenade4 | Smoke |
| Zeus | Zeus |
| MainGameCondition | BuyMenu |
| WeaponChoiceCondition | True |
| TCZ75orTec9 | TCZ75 |
| TDeagleORr8 | TR8 |
| Tmp5SDorMP7 | TMP5SD |
| CTuspsORp2000 | CTUSPS |
| CTDeagleORr8 | CTDeagle |
| CTCZ75or57 | CT57 |
| CTmp5SDorMP7 | CTMP7 |
| CTM4a4orM4a1s | CTM4A4 |
| SideChoice | TSide |
| PistolsCondition | False |
| MolotovBuyCondition | False |
| DecoyBuyCondition | False |
| FlashBuyCondition | True |
| SmokeBuyCondition | False |
| HEBuyCondition | True |
| LimitGrenadeCondition | False |
| PistolNullCondition | True |
| KillNumber | 2 |
| KillEntryNumber | 0 |
| HMKcondition | False |
| BombPlant | True |
| buttonBombPlant | False |
| buttonSuicideReward | True |
| buttonGrenadeUse | False |
| buttonTeamKill | False |
| buttonSurvive | True |
| loseBonusCount | 1 |
| winAfterPistolRound | False |
| howDidYouWinTheRound | Win by team elimination |
| incendiaryBuyCondition | False |
| kits | Kits |

Queries used:

|  |  |
| --- | --- |
| Query | Function |
| SELECT | Returns the result set of the table |
| UPDATE | Changes the data in the table |

**Side choice menu**

Diagram

Description automatically generated

* Player can choose the side by pressing the button and the choice will be saved in the database.
* The choice of side affects the weapon loadout and the aspects of the "Game" and "Result" stage

**Main menu**

Diagram

Description automatically generated

* Allows the user to go through every stage of the game (the next stage cannot be accessed without the previous one, e.g. "Game" before "Buy").
* The MainGameCondition in the database has a value corresponding to the specific stage of the game.
* The game indicators, such as CT & T won rounds, Round count and Balance present on every menu in the game. They are connected to MySQL and update at every stage of the game.

**Buy menu**

Diagram

Description automatically generated

* In the first round, when the Buy Menu is firstly accessed, user can only go to Weapon Choice menu, as it will set the specific weapons loadout based on user preferences.
* After accessing the Weapon Choice menu the user can go to three different menus or go back using the “Go back” button.
* The “Go Back” button after being pressed will lead back to Main Menu, so change the MainGameCondition value, so the Game button can be accessed.

Diagram

Description automatically generated

**Weapons choice menu**

Diagram

Description automatically generated

* The user must select 1 weapon from each pair using the buttons.
* The weapon selection will affect the weapons available to the player later on in the game.
* Go back button is activated as soon as all the weapons are chosen and lead to buy menu.
* There is a choice of weapons for both sides, as the player after the first half of the game (15 rounds played) will move to the other side.

**Weapons menu**

Diagram

Description automatically generated

* Loads data on previously selected weapons, current weapons in inventory and balance. The weapons shown in this menu depend on the previously selected weapon and the selected side.
* The button of the specific weapon is deactivated when:
  + There is not enough money on balance
  + Another weapon was chosen in weapon choice menu
  + You are playing for a side where this weapon is not available
  + You already have this weapon, so you cannot buy it twice (unless you drop it using another menu)
* The balance, chosen weapons, etc. will be updated in the database as soon as user presses the Go back button.

**Weapon drop/pick menu**

Diagram

Description automatically generated

**Pick menu (For Buy Menu)**

Diagram

Description automatically generated

* All possible weapons that can be picked up using buttons are shown.
* For the pistol round (first round of the game) the special requirements are needed, such as:
  + As teams could not pick up the enemy’s weapons, as they have not met in the game yet, the weapons available specifically for the opposing side must not be accessible.
  + In the first round it is impossible to buy heavy weapons, so they must not be available as well.
* In the menu the current inventory is shown. In case of picking a weapon, this list is updating in real time. As soon as the weapon is chosen, the weapons field values are updating in database.

**Drop menu (For Buy Menu)**

Diagram

Description automatically generated

* A user can drop any weapon using the buttons.
* As soon as the weapon is chosen, the weapons field values are updating in database.

**Game menu**

A picture containing line chart

Description automatically generated

* The user can talk about how the round is going by using the buttons.
* All the buttons before pressing “How many kills?” button are disabled.
  + They become enabled after the Kill Menu have been visited OR "how many kills?" input equals to 0.
  + Pressing the “How many kills?” button will bring up the input menu, where the user is required to enter the kill number between 0 and 5.
  + As soon as the user entered the number, the Kill Menu button becomes enabled, so the user can access the kill menu.
  + After pressing "How many kills?" button , the remaining buttons become enabled.
* Each button pressed, whether it has an input field or not, changes the balance indicator depending on input by user.
  + The data whether the user planted the bomb or not will be stored in MySQL, as it can be used in the future in the Results stage of the game.
  + For example, pressing the "Did you use grenades?” button will show the input field, where the user can enter Y/N whether he used grenades or not.
    - If entered Y, all the others buttons will be disabled and the available grenades buttons will appear on the screen.
    - If the user presses on this buttons, they will be “used”, so in their field in MySQL their value will be NULL.
    - As soon as user is finished with grenades, he can press the button “Finish with grenades”, which will activate the other buttons, hide the “grenades” buttons and disable the “Did you use grenades?” button.
* Side-dependent in-game conditions, such as “Did you plant the bomb” are available for the user based on the side he is currently playing for.
  + This data is loaded from database
* Go back button changes the Main Game Condition in MySQL to “Result Menu” value, allowing to access the Result Menu from the Main Menu.

**Kill menu**

Diagram

Description automatically generated

* 5 kill buttons, which lead to the specific kill menu.
* Each button is activated according to the input of number of kills in the previous menu, stored in database.
* After pressing the Kill button the user is directed to the specific kill menu, where he can choose the way how he killed an opponent.
  + Than the user comes back to Kill menu, so the next kill button is activated, if there is more than 1 kill.
* Between these kills the user may have dropped/picked a weapon, so the button to go to the drop / pick menu is available.
* The Go Back button directs back to the result menu and inactive until all the kill menus are accessed.

**Specific kill menu**

Diagram

Description automatically generated

* The user can choose with which weapon did he make a kill.
  + It is important as each weapon has different kill award, which changes the player’s balance.
* The balance updates as soon as the button was pressed.
* If the kill was done using the grenade, the value of this grenade will become NULL in database, as it was used.
* The weapons available in this menu are loaded from database based on previous user actions.
* The Go Back button becomes enabled only when the weapon was chosen.
  + When pressing the Go Back button, the data about weapons and balance is loaded to MySQL database.

**Pick menu (For Game Menu)**

Diagram

Description automatically generated

* The pick menu is the same as for the Buy stage of the game, but now during the pistol round the enemy-exclusive weapons can be picked up.

**Result menu**

Diagram

Description automatically generated

* The user can state the results of the round.
* The “Did you survive button?” allow user to input whether he survived in the end of the round or not.
  + If player did not survive, than the values of the ammunition fields in database will be NULL
  + The primary weapon will be set to default one, depending on the side
* The answer to “Did you win the round?” button will determine which buttons will appear in the menu:
  + The way how you won the round
  + The way how you lost the round
    - Each of the ways have different rewards, so it will change the player balance.
* As soon as all the required buttons are pressed, the Continue button will appear on the screen.

**Final result menu**

Diagram

Description automatically generated

* The player can see:
  + Which weapons has he saved for the next round
  + How did he win/lost the round
  + What is the score and the round count
  + The balance he has by the end of the round
* The Continue button set the Main Game Condition to Buy Menu
* The menu will display “the end of the game message” if:
  + The 30 rounds were played
  + One team have won 16 rounds
  + The score is 15-15 (a draw)

# Key algorithms

# Weapon Choice

Diagram

Description automatically generated

class WeaponChoice() //class of a menu where you can choose a weapon

STATEMENT DatabaseImportstmt //opens a statement, which allows to work with the database

SELECT DatabaseImport //selects needed data from database

resultSet WeaponChoiceCondition //condition, stored in the database, which enables the weaponPurchase menu, if TRUE. Initially is FALSE

end SELECT

end STATEMENT

integer WeaponchoiceCount = 0 //The counter for the class. Increases each time when the weapon is chosen

String Weapon1orWeapon2string //The string will get the value of the chosen weapon between 2 options, so it can be updated to database

String Weapon3orWeapon4string

...

String Weapon7orWeapon8string

button Weapon1Choice //Allows the user to choose the weapon by pressing the button

button Weapon2Choice

...

button Weapon7Choice

button Weapon8Choice

button QuitMenu //button that allows to come back to buy menu. As after pressing this button no actions can be taken, it used for   
 //updating the data in database

QuitMenu.disable() //the quit button should be initially disabled, as some actions needed to be taken before quitting

if Weapon1 = action.pressed then //if the Weapon1 button is pressed, therefore the user preferred Weapon1 over Weapon2

Weapon1Choice.disable() //disables the pressed button, and the button for the opposite weapon

Weapon2Choice.disable()

String Weapon1orWeapon2string = "Weapon 1" //The string gets the value of the chosen weapon, so it can be updated to database

WeaponchoiceCount = WeaponchoiceCount + 1 //the counter increases each time when the choice between two weapons is made

end if

if Weapon2 = action.pressed then //if the Weapon2 button is pressed, therefore the user preferred Weapon2 over Weapon1

Weapon1Choice.disable()

Weapon2Choice.disable() //disables the pressed button, and the button for the opposite weapon

String Weapon1orWeapon2string = "Weapon 2" //The string gets the value of the chosen weapon, so it can be updated to database

WeaponchoiceCount = WeaponchoiceCount + 1 //the counter increases each time when the choice between two weapons is made

end if

...

if Weapon8 = action.pressed then //if the Weapon8 button is pressed, therefore the user preferred Weapon8 over Weapon7

Weapon7Choice.disable()

Weapon8Choice.disable() //disables the pressed button, and the button for the opposite weapon

String Weapon7orWeapon8string = "Weapon 8" //The string gets the value of the chosen weapon, so it can be updated to database

WeaponchoiceCount = WeaponchoiceCount + 1 //the counter increases each time when the choice between two weapons is made

end if

if WeaponchoiceCount = 8 then //as there are 8 pairs of 16 weapons, therefore when the variable equals to 8, all the choices are made

QuitButton.enable() //enables the quit button

end if

if QuitButton = action.pressed then

STATEMENT DatabaseUpdate //opens the statement, which allows to update the database

query UPDATE Database //the query to update the database

WeaponChoiceCondition = TRUE //sets the condition equal to TRUE, which allows to access the WeaponPurchase menu in BuyMenu

Weapon1or2Database = Weapon1orWeapon2string //saves the value of the string in the database, so only one of two weapons will

//be available in the WeaponPurchase menu

Weapon3or4Database = Weapon3orWeapon4string

...

Weapon7or8Database = Weapon7orWeapon8string

end query

end STATEMENT

open BuyMenu() //opens the Buy Menu

close WeaponChoice() //closes the WeaponChoice Menu

end if

end class

# Weapon purchase

Diagram

Description automatically generated

class WeaponPurchase() //class of a menu where you can buy a weapon

STATEMENT DatabaseSelectstmt //opens a statement, which allows to work with the database

SELECT DatabaseImport //selects needed data from database

resultSet Side //selects the side that the player have chosen for the game

resultSet Weapon1or2Database //selects the weapon choices from the previous class

resultSet Weapon3or4Database

...

resultSet Weapon7or8Database

resultSet Weapon1Inventory //selects the weapon that the user currently has in his inventory

resultSet Weapon2Inventory

resultSet Grenade1Inventory //selects the grenades that the user currently has in his inventory

...

resultSet Grenade4Inventory

resultSet ArmorInventory //selects the type of armor that the user currently has in his inventory

resultSet fullArmor //selects whether the user has the full armor (not damaged) in his inventory or not

resultSet zeusInventory //selects whether the user has zeus in his inventory or not

resultSet kitsInventory //selects whether the user has kits in his inventory or not

resultSet Balance //selects the current balance of the player

resultSet RoundsWonT //selects the number of rounds that was won by the T side

resultSet RoundsWonCT //selects the number of rounds that was won by the CT side

resultSet RoundsCount //selects the number of rounds that were played overall

resultSet MolotovBuy //checks whether the grenade was bought in this round or not (when coming back to buy menu)

resultSet HEGrenadeBuy

...

resultSet SmokeBuy

end SELECT

end STATEMENT

Text Field TFWeapon1 = set text Weapon1Inventory //text fields display the items in the inventory

Text Field TFWeapon2 = set text Weapon2Inventory

Text Field TFZeus = set text zeusInventory

Text Field TFGrenade1 = set text Grenade1Inventory

Text Field TFGrenade2 = set text Grenade2Inventory

...

Text Field TFGrenade4 = set text Grenade4Inventory

Text Field TFKits = set text kitsInventory

Text Field TFArmor = set text ArmorInventory

Text Field TFRoundCount = set text Weapon1Inventory //text fields display the information about the game

Text Field TFBalance = set text Balance

Text Field TFRoundsWonT = set text RoundsWonT

Text Field TFRoundsWonCT = set text RoundsWonCT

label lSide = set text Side //Displays the side that the player is playing for

button Weapon1NameBuy //buttons to buy a weapon

button Weapon2NameBuy

button Weapon3NameBuy

...

String Weapon1NameString //the name of the purchased weapon to be stored in mySQL

String Weapon2NameString

...

button QuitWeaponPurchaseMenu //allows to quit the weapon purchase menu

if Grenade1Inventory.equals(not null) && ... Grenade4Inventory.equals(not null) then //statement used to disable the buttons used to purchase any grenades, as 4 is the maximum number of the grenades in the inventory

button Molotov.disable()

...

button HEGrenade.disable()

end if

if Side = TSide then //enables only the buttons for weapons available for the T side

enable buttons weapons for T Side

disable buttons weapons for CT Side

enable buttons weapons.(weaponChoiceMenu) //enables only the buttons for chosen weapons

end if

if Side = CTSide then //enables only the buttons for weapons available for the CT side

enable buttons weapons for CT Side

disable buttons weapons for T Side

enable buttons weapons.(weaponChoiceMenu) //enables only the buttons for chosen weapons

end if

if balance < weapon1price then //if the player does not have enough balance to purchase a specific weapon, disable the button

button Weapon1NameBuy.disable()

end if

...

if MolotovBuy = "True" then //disables the button of the grenade that was purchased before

button Molotov.disable()

end if

...

if Weapon1NameBuy = action.pressed then //the weapon was purchased using the button

Balance = Balance - weapon1price //the balance updates

TFBalance = set text Balance //the text field updates

Weapon1Inventory = Weapon1NameString //the database variable updates with the value of the string of the purchased weapon

Weapon1NameBuy.disable() //disables the button, as it was already pressed

end if

...

if QuitWeaponPurchaseMenu = action.pressed then

STATEMENT DatabaseUpdate //opens the statement, which allows to update the database

query UPDATE Database //the query to update the database

Weapon1Inventory //updates the weapon variable in the database

...

Balance //updates the balance in the database

end query

end STATEMENT

open BuyMenu() //opens the Buy Menu

close WeaponPurchase() //closes the WeaponChoice Menu

end if

end class

# Testing Plan

|  |  |  |
| --- | --- | --- |
| Success criteria | Testing method | Expected outcome |
| All the menus should have buttons. Every menu should have quit/continue button, if needed. The buttons should not be able to press should be disabled. | Running the program and checking whether it is possible to press the buttons. | The menus should have buttons which the user should be able to press if the conditions are met. The conditions should be loaded to MySQL database previously and downloaded from there later. When pressing them, they should transfer user to another menu, to call an input menu, or to change data in the game, for example, balance. |
| Every menu, expect for the Side Choice should have Round Count, Round won for T/CT and Balance fields. The data should be downloaded from the MySQL database. | The data in the text fields should correspond with the data in the MySQL, so the correctness of the data can be checked in the database. | The text fields should show the actual information about the game: Round Count – how many rounds played; Rounds T – how many rounds have the T side won; Rounds CT – how many rounds have the CT side won; Balance – what is the player’s current balance. |
| Calculations associated with balance. Update and display the balance according to the events of the game. | As the balance changes in the program must match the events in the game, the software can be run in parallel with the game and check the balance values for consistency. This can also be compared to the game's official website. | The events associated with balance are purchasing weapons and receiving bonuses (positive in most cases, negative in one case). In case of purchasing weapons, the weapons’ buttons, which cost more than the player’s current balance should be disabled. Also, after player is buying a weapon, the new balance must be calculated using the formula:  New balance = Balance – Price of the weapon.  In case of receiving bonuses, the player receives different bonuses for in-game events. For example, the reward of kill is changing depending on the weapon used, which also is needed to be taken into account. Also, there is always a different reward for different ways of winning and losing rounds. This data should be saved in the MySQL and displayed in the balance text field. |
| Quit and continue buttons. Buttons should be switched on at the appropriate moment, in order to allow the player to move from one menu to another. The conditions should be loaded to and taken from MySQL and being taken into account. | Using the program and checking whether the quit and continue button become available at the appropriate time. | Allowing the user to use the quit and continue buttons, so they can move between different menus. Also, as pressing these buttons indicate that the user is not going to use the menu anymore, therefore the data is not going to change, so these buttons can be used as data-loading buttons in MySQL. |
| Text boxes displaying the weapons purchased. | Buying weapons in the Buy Menu and checking whether they are displayed in the text fields. | The text fields will show the weapons purchased by the player. Done to show to the player which weapons he has in his inventory. |
| The playing conditions correspond to the user's choice. For example, the choice of side. | Make certain choices and check whether further choices are in line with the player's choices. | The choices made at the beginning of using the programme correspond to further events. For example, if the user in the Weapon Choice Menu has selected one of the two weapons, then in the Buy Menu only one of these weapons should be available for purchase. Also, in Result Menu offered options to win a round should correspond to the choice of the side. For example, only the "Win by bomb detonation" option should be available for the T Side, and the "Win by bomb defusal" option for the CT Side. |
| Display the final results in the last menu. | Complete walkthrough of the program and check the results displayed. Verification can be done by comparing with real game data. Also, the displayed data must match the data in MySQL. | The last menu should display the results of the round. These results include: the weapons remaining with the player at the end of the round, the balance, the round count, the number of rounds won by both sides, and how the side the player plays for won or lost the round. This data has been loaded into MySQL previously and will be loaded from MySQL in this menu. |
| The program can run in a kind of "loop" before the game comes to a logical end. | Checks that the program works without errors until one side wins 16 rounds, or 30 rounds have elapsed in the game (the game comes to an end). | As the main purpose of the program is to calculate the spending during the game, it must be able to calculate it for all 30 rounds without any errors. To do this, after each round some conditions must be reset to the original conditions. For example, the conditions regarding button presses on certain menus - these must be reset before each new round. Also, some conditions must be changed during the game, as after the 15th round the player moves to the other side. |
| The user should be able to choose a side – T or CT | Choosing a side in the first, side choice menu, by pressing the button and check that the further options in the program correspond to that choice. | After selecting a side, the main menu should have a label that says which side the player is playing for. In the weapons purchase menu, only weapons that are available to the selected side should be available to the player. In the round result menu the user should only have the win/loss options that correspond to the chosen side.  For example, if the user has selected the T side, the label should write a phrase in the main menu along the lines of "You are playing for the T side". In the purchase menu the user must have a weapon available for the T side, such as the Galil AR, but not a weapon available for the CT side, such as the M4A4. In the final menu, he should have the option to win with a bomb tab, as this is exclusive to the T side, while not having the option to win with a bomb defusal, as this is exclusive to the CT side. |
| Main menu with three buttons: Buy, Game, Result. Each button represent the stages of the round: Buy - what do you want to buy; Game - how is the game going (any kills, was the bomb planted); Result - did you win the round, etc. They have specific order: you cannot access Game and Result, if you haven’t pressed Buy, etc.  Pressing each button should open a corresponding menu. | Go to Main Menu and press the Buy, Game and Result buttons in order to check whether they are working or not: do they work in the correct order (Buy – Game – Result), do they open the correct menus? | When opening the Main Menu, the three buttons should be there: Buy, Menu, Result. In the beginning, only the Buy button should be enabled. After pressing the Buy button, the Buy menu should open. After coming back from the Buy menu to the Main menu, the Buy and Result buttons should be disabled and the Game button should be enabled, pressing which should open the Game menu, etc. |
| Buy Menu:  Weapon choice menu.  In this menu the user should be able to choose between the interchangeable weapons. Some weapons in CS:GO cannot be used in the same game, so you need to choose between them, for example, Desert Eagle or R8 Revolver. Does not apply for all the weapons. After everything is chosen, the button will be disabled till the end of the game. Only works in the first round. | Go to weapon choice menu, choose weapons, then go to the weapon purchase menu, and check that the buttons of the selected weapon are enabled and the buttons of weapons that have not been selected are disabled. | There are a total of 8 pairs of weapons in the game, between which one of the two must be selected. Once one weapon of the pair is selected, in the Weapon Choice Menu one weapon of the pair must be switched on while the other is switched off.  For example, if the user selected the M4A4 in the Weapon Choice Menu instead of the M4A1-S, then the button to buy the M4A4 must be enabled in the purchase menu, while the button to buy the M4A1-S must be disabled. |
| Buy Menu:  Weapon purchase menu. The user should be able to purchase weapons using buttons. | Go to the Weapon purchase menu and press the button of the specific weapon to check whether it was “bought” or not. Specific game indicators will show whether the purchase was successful. | The user should be able to buy weapons at the push of a button. Once the button is pressed, the weapon must be "purchased". This must be accompanied by certain events. First, the button for purchased weapons must be disabled, because the purchase has already been made and the user cannot buy the same weapon a second time. Second, the balance must decrease according to the formula (new balance = balance - price of the weapon). The new balance should be displayed in the balance text box. Third, the text in the textbox of the current weapon in the inventory must be updated to show the name of the purchased weapon instead of null or the previous weapon. Fourth, after exiting this menu, the data about the purchased weapon and the updated balance should be loaded into the database.  For example, when the AWP button is pressed, the AWP button must be disabled, the balance must decrease by 4750 (price of AWP), the AWP text must be in the weapon text field and when exiting the program, the AWP text must be in the given weapon type field in the database. |
| Buy Menu & Game Menu:  Weapon drop/pick menu.  The user should be able to pick and drop the weapon. | Go to weapon pick and drop menu and press the buttons for the weapons that I wish to pick or to drop. The in-game indicators should change as well as the data in the database. | When the user presses the button to "pick" a weapon, the text field with the current weapon must be updated with the text of the corresponding weapon and the text of this weapon must be added to the corresponding weapon type field in the database. When the user "drops" a weapon, the text field of the corresponding weapon type must be empty (or write "null") and the field in the database must also be empty.  For example, if a user decides to press the "SSG 08" button in the pick menu, the text "SSG 08" should be in the inventory text field, as well as in the corresponding field in the database. If the "MP9" button in the drop menu is pressed, the text field and the field in the database of the respective weapon type becomes blank. |
| Game menu & Result Menu:  The user should be able to enter data about events occurring in the round using buttons and text input. | Go to Game Menu and press the buttons that are asking about the game events and answer the questions by pressing the buttons and typing and check whether the results correspond to the real game. | The user should be able to answer questions about the game using buttons and text boxes. Depending on their choice, the balance and the results of the round should change.  For example, if the user presses the "Did you plant the bomb?" button, a text entry menu will pop up where the user has to enter Y/N (Yes or No). If the user enters Y, their balance will increase by $300. |

Word count: 0