Assessment 2 Design

# Project description

This project aims to create a clicker game with a currency, upgrades, enemies, health, restarting and cheats.

* Health is a decimal that resets and increases the points when it reaches 0.
* Points are a decimal that shouldn’t be able to go below 0 and increase when the wave ends.
* There must be onscreen zones for showing scores.
* Upgrades are available to purchase that auto click at higher rates.
* There must be a way to restart the game.
* Support for cheats.

There is also a part 2 that also requires…

* A boss fight every now and then.
* Coins are constantly falling if the boss cookie is alive.
* The boss is unaffected by DMG and DPS, instead you must sacrifice upgrades in order to get specialized DMG

Table of Contents

[Project description 1](#_Toc517341253)

[CDT 3](#_Toc517341254)

[CDT 3](https://livenmitac-my.sharepoint.com/personal/asriel-repp_live_nmit_ac_nz/Documents/Documents/SDV/Project%202%20Design.docx#_Toc517341255)

[Overview 4](#_Toc517341256)

[Narrative 1 4](#_Toc517341257)

[Action list 1 4](#_Toc517341258)

[Narrative 2 5](#_Toc517341259)

[Action List 2 5](#_Toc517341260)

[Narrative 3 5](#_Toc517341261)

[Action List 3 5](#_Toc517341262)

[DDT 6](#_Toc517341263)

[Prototype 1 scope 6](#_Toc517341264)

[Narrative 6](#_Toc517341265)

[Test 6](#_Toc517341266)

[Prototype 2 scope 6](#_Toc517341267)

[Narrative 6](#_Toc517341268)

[Test 6](#_Toc517341269)

[Prototype 3 scope 6](#_Toc517341270)

[Narrative 6](#_Toc517341271)

[Test 7](#_Toc517341272)

[Prototype 4 scope 7](#_Toc517341273)

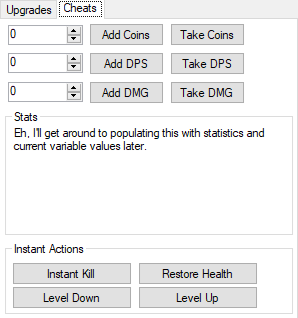
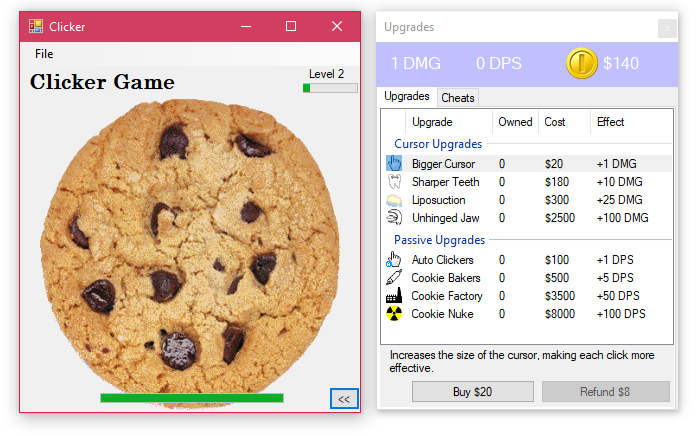
[Narrative 7](#_Toc517341274)

[Test 7](#_Toc517341275)

[Questions and Answers 7](#_Toc517341276)

Planning

# CDT



## CDT

|  |  |  |  |
| --- | --- | --- | --- |
| **Control Type** | **Type** | **Events** | **Comments** |
| imgCookie | Image | Mouse Down | HP down, coins up, cookie image has a bite taken out of it. |
| pbHP, pbXP, lblLVL | Progress Bar, Label | - | Shows HP and XP |
| lblCoins, lblDMG, lblDPS | Label | - | Shows coins, DMG and DPS |
| btnBuy, btnRefund | Button | OnClick | Buys or refunds selected upgrade -> coin |
| lstUpgrades,  lblDescription | ListBox, Label | OnClick | Lists all upgrades and counts how many you have purchased. |
| num(Add/Take)  (Coins/DPS/DMG) | Number, Button | OnClick | Add or Remove value in the num to the Coins, DPS or DMG variable. |
| btnInstaKill, btnRestoreHP, btnLVLDown, btnLVLUP | Buttons | OnClick | Changes values of related variables. |
| lblStats | Label | - | Statistics about the gameplay so far. |

# Overview

The user is presented with a cookie to click on, as time progresses, the cookies and leveling up become exponentially harder to do, and every 5 levels, there is a boss cookie. Finding the most efficient tactics for leveling up and increasing your attack power is the name of the game. Along with surviving the Boss’ mechanics of draining Coins and requiring upgrade sacrifices in order to do damage.

# Narrative 1

The user can click on the cookie to increase their score and decrease the health of the cookie (the cookie has 10 versions of the image that represent HP, it will cycle through them).

If the user wants to use cheats, there’s a second tab in which options for changing the mechanics of the game, including artificially adding to, or taking from, the Coin or CPM variables, instantly killing or healing the cookie, or leveling up or down are possible.

# Action list 1

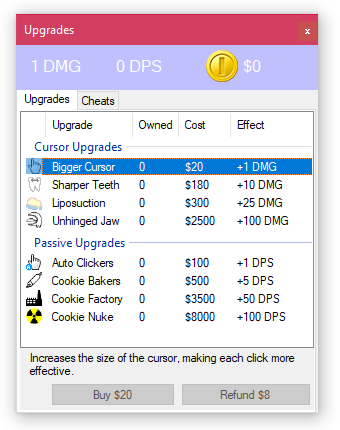
1. On initialization, all variables are initialized with default values.
2. frmMain and frmUpgrades appears.
3. A timer is set up to go off every second. [DPSTimer]
   1. When triggered, it reduces mdHP by mdDPS [UpdateHP()]
   2. The timer also moves frmUpgrades so it’s always next to frmMain.
4. The user clicks on imgCookie
   1. mdHP is reduced by mdDMG. [UpdateHP()]
5. UpdateHP() is called.
   1. The Cookie image is changed to an appropriate image for its current HP.
   2. The HP progress bar is also updated.
6. The user uses a cheat.
   1. The cheat does as the button describes.

# Narrative 2

Once HP of the cookie reaches 0, HP is reset, XP is increased and potentially leveled up.

# Action List 2

1. UpdateHP() is called.
   1. The Cookie image is changed to an appropriate image for its current HP.
   2. If mdHP is below or equal to 0, now it’s reset to LVL^2 \* 5 and HPcap is also set to the same value , the lcHPcap is also added to mdXP. [UpdateXP()]
   3. The HP progress bar is also updated.



# Narrative 3

Once they have enough Coins, they can buy upgrades. The upgrades are sorted by utility and more expensive ones have more effectiveness, the descriptions and specifications of upgrades can be found by clicking on them and reading the description below.

If the user intends on refunding an upgrade, they can receive half of the value of the upgrade in return when clicking btnRefund.

# Action List 3

1. The user selects an Upgrade
   1. The description for the upgrade is changed.
   2. The values in the Buy and Refund buttons are updated in [UpdateCoins()]
2. The user clicks on btnBuy or btnRefund
   1. Checks to see if the cost or count of the selected upgrade is high enough for that to be possible. The price is multiplied or divided by 1.5.
   2. If it is, the mdDMG and mdDPS are increased or decreased depending on the properties of the upgrade. (This information is read directly from the listview)
   3. The mdCoins are updated and so are btnBuy and btnRefund. [UpdateCoins()]
3. UpdateCoins() is called.
   1. The mdCoins indicator is updated.
   2. btnBuy and btnUpgrade are updated to show the cost of the current upgrade. They’re also disabled if you don’t have enough count or Coins to press the buttons.
4. UpdateDPS() or UpdateDMG() are called.
   1. They only update the relevant indicators on screen.
   2. UpdateXP() is called.
   3. If XP is greater than LVL \* 100, LVL is increased.

# Narrative 4

The boss fight mechanic forces users to sacrifice upgrades in order to become powerful enough to take out the boss cookie. The cookie takes coins every second, so if the user isn’t prepared, they might lose the game, in which case a game over screen appears and shows you your statistics from the playthrough.

# Action List 4

1. The Boss Cookie Spawns (by hiding frmMain and showing frmBoss, this also hands over control of frmUpgrades)
2. Every second, a timer goes off, which takes away coins.
3. On boss cookie click, pbHP is reduced. [frmBoss.UpdateHP()]
4. frmBoss.UpdateHP() is called.
   1. The change to HP is showed on the progress bar.
   2. If the health is less than or equal to 0…
      1. The user is notified and rewarded coins.
      2. frmMain is restored and given back full control of frmUpgrades.
      3. frmBoss is destroyed.
5. frmUpgrade handles sacrificing upgrades in order to increase Boss DMG.
6. In the event the player loses the fight with the Boss Cookie (by being unable to take any more coins), frmMain handles ending the game, which involves unloading all forms and showing frmEndGame.

# DDT

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Range | Purpose |
| mdHP, mdHPCap | Integer, UInteger | -2,147,483,648 to 2,147,483,647 | Store current HP of the cookie and it’s max HP |
| mdXP, mdCoins | UInteger | 0 through 4,294,967,295 | Store XP and number of Coins available to spend. |
| mdLVL, mdDMG, mdDPS | UShort | 0 through 65,535 | Store the level, Damage and Damage per second. |
| pbHP, pbHPCap, pbDMG | Single | -3.4028235E+38 through -1.401298E-45 for negative values  1.401298E-45 through 3.4028235E+38 for positive values | Boss Cookie’s own set of variables. They’re singles so that decimal places are possible. |

Everything else is fetched from rows in listUpgrades, there is also listStats, which is updated via various handles throughout the program.

Prototypes

# Prototype 1 scope

Narrative

Variables are initialized, any buttons the user presses should update variables in code and change what is presented on screen.

## Test

|  |  |  |
| --- | --- | --- |
| Progress Bars | Labels | Upgrades Tab |
| XP and HP scroll whenever updated. | DMG, DPS, Coins, and LVL all show correct values when updated. | Selecting an upgrade and buying or refunding one updates the Owned column, DMG, DPS and Coins, updates lblDescription and enables or disables the Buy and Refund buttons. |

# Prototype 2 scope

Narrative

The user can click on the cookie and it should work as expected.

## Test

|  |  |  |
| --- | --- | --- |
| Dealing Damage | Killing a Cookie | Leveling Up |
| HP Bar goes down. No overflow error on HP going below 0. | HP Bar is reset, XP bar goes up. | XP bar is reset, lblLVL is updated and new cookies have a higher difficulty. |

# Prototype 3 scope

Narrative

The user can buy/refund upgrades and have their effects apply.

The Damage per Second mechanic needs to be working.

## Test

|  |  |  |
| --- | --- | --- |
| DPS | DMG | Buy/Refund Upgrade |
| Every second, HP is taken down by DPS. No overflow error on HP going below 0. | Every click, HP is taken down by DMG. No overflow error on HP going below 0. | Buying/Refunding increases/decreases the Owned Column, DPS, DMG, Coins and all changes are shown on screen. |

# 

# Prototype 4 scope

Narrative

The boss cookie mechanic completely works.

## Test

Level up to level 5, fight the boss and win, level up to level 10, lose to the boss and see the endgame screen.

# Questions and Answers

**What modules do you see that could come in handy to reduce the amount of code you are writing?**

Perhaps a module that manages storing all the variables and updates Labels on screen when they’re updated.

**What functions or subtasks could be used that would be reusable and useful?**

The Update…() functions I made are pretty useful.

**What 3 data types would you consider for the values we are using and why?**

Integer, UInteger and UShort. In testing I found that Shorts aren’t long enough for HP, HPcap, XP and Coins. These values get the highest the fastest and often cause overflow errors in the late game.