Combat Robot

Data Structures Group 5 Final Project Report

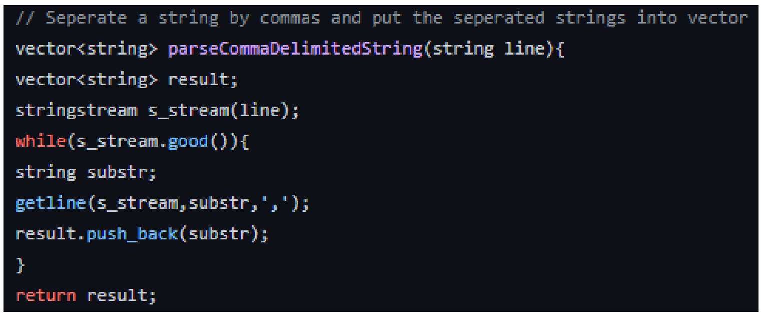
Darryl Maulana Hilmy – 2502009652
Bayu Hartho Leksono – 2502013731
Arvin Yuwono - 2502009721

Problem Description

This project is a crude adaptation of Hero of Robots, a Taiwanese arcade trading game. It simulates where upon starting the game, the user acquires a new card which allows them to play with it or any other cards the user owns against enemies. This iteration is the best possible impression coded in C++, and the prototype has been proven to be partially successful, with some subsequent errors occurring occasionally.

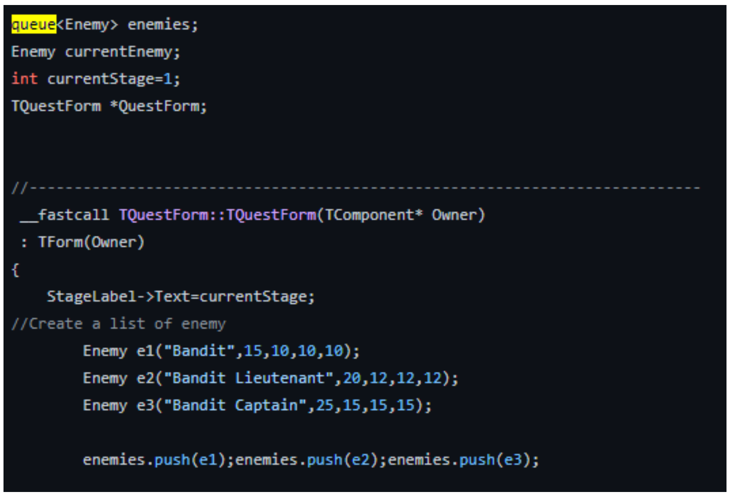
DataStructures Used

- Vector
  Separates a string via commas and places the separated strings into a vector. Using data it reads from robots.txt, the values are used to create objects of the type Robot that are later on ushered into the vector. A vector of the type Robot is initialized with the aforementioned function.



```
// Seperate a string by commas and put the seperated strings into vector
vector<string> parseCommaDelimitedString(string line){
vector<string> result;
stringstream s_stream(line);
while(s_stream.good()){
string substr;
getline(s_stream,substr,',');
result.push_back(substr);
}
return result;
```

- Queue
  Represents the order of the enemies encountered by the player and are placed in a queue to ramp up the difficulty of the simulation.

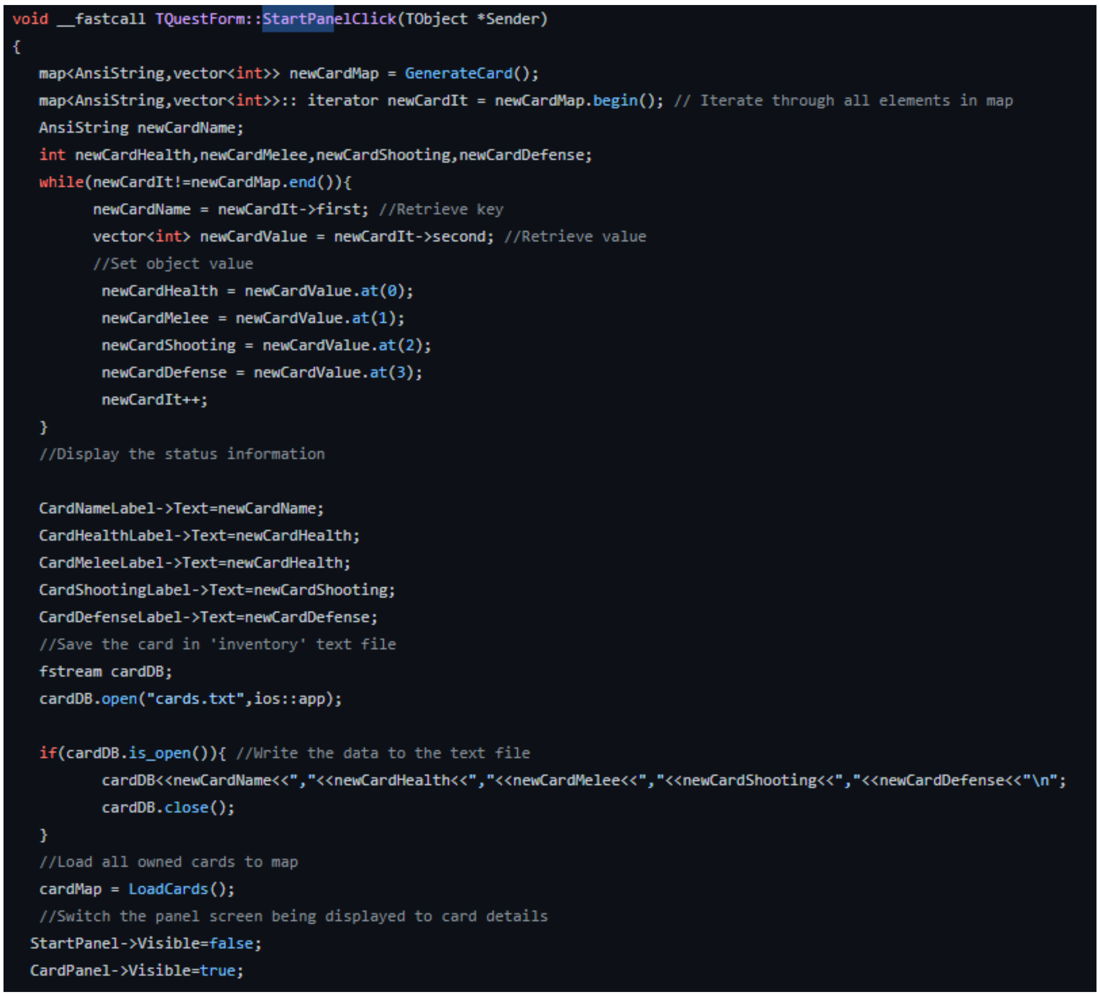


```
queue<Enemy> enemies;
Enemy currentEnemy;
int currentStage=1;
TQuestForm *QuestForm;


//---------------------------------------------------------------------------
 __fastcall TQuestForm::TQuestForm(TComponent* Owner)
 : TForm(Owner)
{
    StageLabel->Text=currentStage;
//Create a list of enemy
        Enemy e1("Bandit",15,10,10,10);
        Enemy e2("Bandit Lieutenant",20,12,12,12);
        Enemy e3("Bandit Captain",25,15,15,15);

        enemies.push(e1);enemies.push(e2);enemies.push(e3);
```

- Map
  A Dictionary-Esque data structure that prevents the occurrences of identical keys. It is used for differentiating between robot names.



```
void __fastcall TQuestForm::StartPanelClick(TObject *Sender)
{
   map<AnsiString,vector<int>> newCardMap = GenerateCard();
   map<AnsiString,vector<int>>:: iterator newCardIt = newCardMap.begin(); // Iterate through all elements in map
   AnsiString newCardName;
   int newCardHealth,newCardMelee,newCardShooting,newCardDefense;
   while(newCardIt!=newCardMap.end()){
         newCardName = newCardIt->first; //Retrieve key
         vector<int> newCardValue = newCardIt->second; //Retrieve value
         //Set object value
          newCardHealth = newCardValue.at(0);
          newCardMelee = newCardValue.at(1);
          newCardShooting = newCardValue.at(2);
          newCardDefense = newCardValue.at(3);
          newCardIt++;
   }
   //Display the status information

   CardNameLabel->Text=newCardName;
   CardHealthLabel->Text=newCardHealth;
   CardMeleeLabel->Text=newCardHealth;
   CardShootingLabel->Text=newCardShooting;
   CardDefenseLabel->Text=newCardDefense;
   //Save the card in 'inventory' text file
   fstream cardDB;
   cardDB.open("cards.txt",ios::app);

   if(cardDB.is_open()){ //Write the data to the text file
          cardDB<<newCardName<<","<<newCardHealth<<","<<newCardMelee<<","<<newCardShooting<<","<<newCardDefense<<"\n";
          cardDB.close();
   }
   //Load all owned cards to map
   cardMap = LoadCards();
   //Switch the panel screen being displayed to card details
  StartPanel->Visible=false;
  CardPanel->Visible=true;
```

Game Description

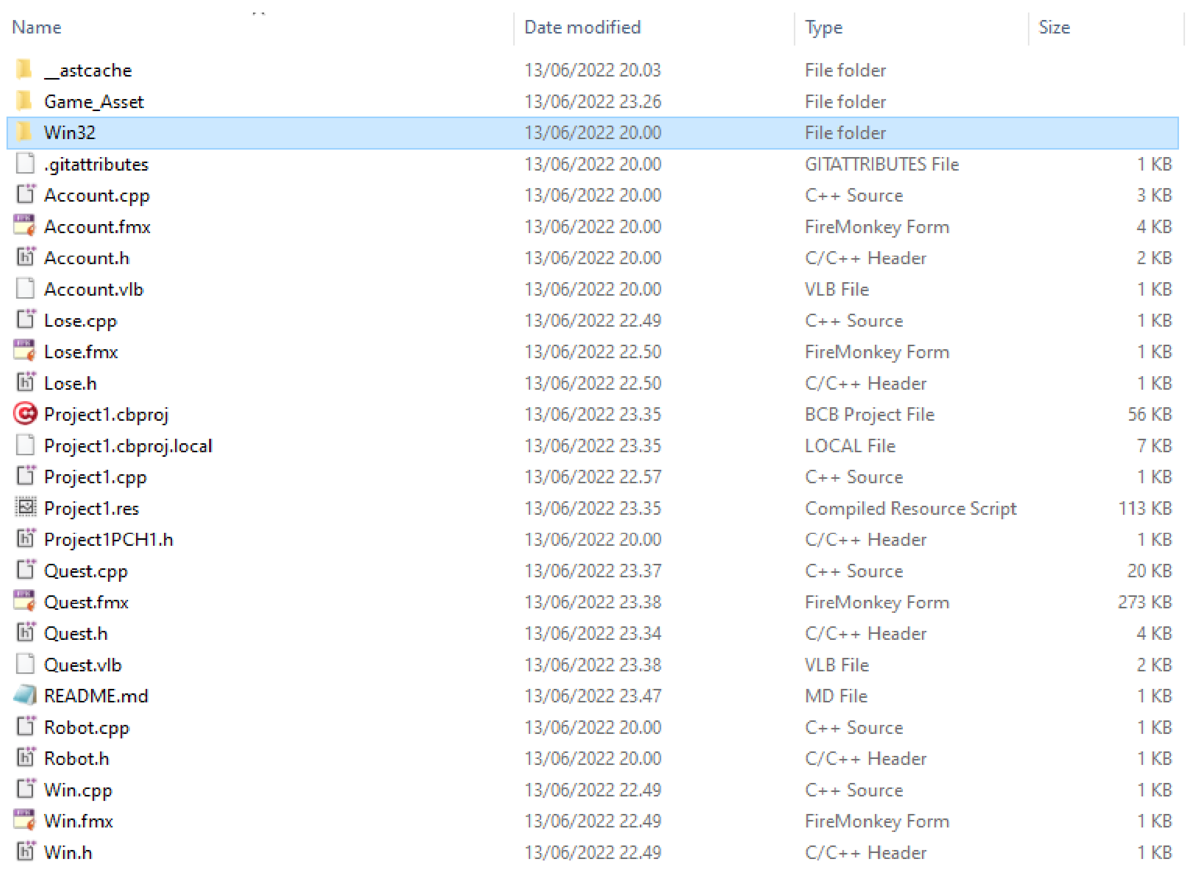
The game begins with the player starting at the login interface, where the player logs into the game. The game will then transition with the player placed into battle and facing waves of AI enemies. The player will be granted an interface and will be able to select between a melee attack or a ranged attack. The enemy will initiate self-defense via an anti-melee or an anti-range. The enemy uses random number generation to determine which defense they will do.

Upon being executed, the decisions made by the player and the enemy will result in numerous cycles of attacking and defending. If the player achieves victory, they will transition to the next stage and recover all lost health points. If the player loses then the game will conclude.

Program manual

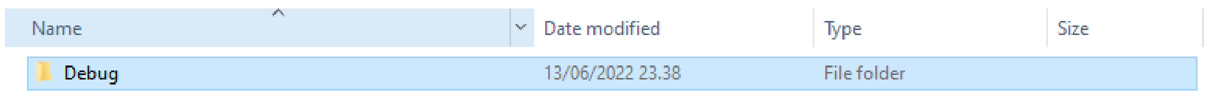
The .exe file is available in the Win32/Debug folder. The code/script responsible for the program can be found in the 'Quest' and 'Project1' files.

Step 1: Open the 'Win32' Folder



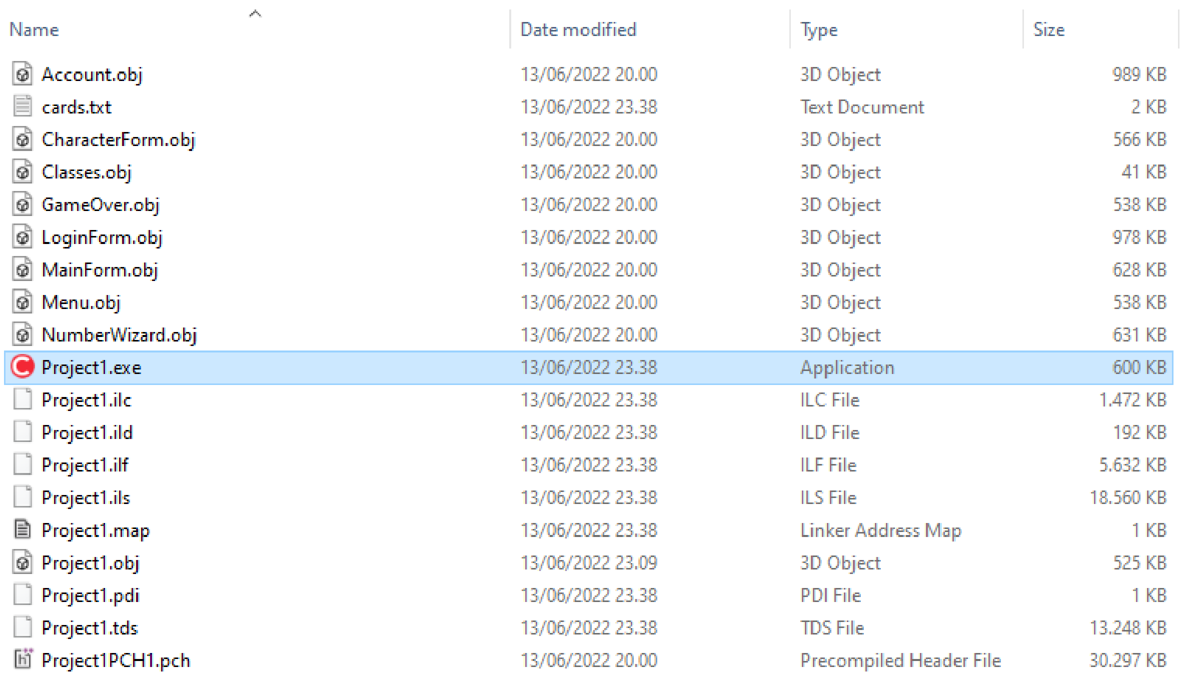
Name	Date modified	Type	Size
__astcache	13/06/2022 20.03	File folder	
Game_Asset	13/06/2022 23.26	File folder	
Win32	13/06/2022 20.00	File folder	
.gitattributes	13/06/2022 20.00	GITATTRIBUTES File	1 KB
Account.cpp	13/06/2022 20.00	C++ Source	3 KB
Account.fmx	13/06/2022 20.00	FireMonkey Form	4 KB
Account.h	13/06/2022 20.00	C/C++ Header	2 KB
Account.vlb	13/06/2022 20.00	VLB File	1 KB
Lose.cpp	13/06/2022 22.49	C++ Source	1 KB
Lose.fmx	13/06/2022 22.50	FireMonkey Form	1 KB
Lose.h	13/06/2022 22.50	C/C++ Header	1 KB
Project1.cbproj	13/06/2022 23.35	BCB Project File	56 KB
Project1.cbproj.local	13/06/2022 23.35	LOCAL File	7 KB
Project1.cpp	13/06/2022 22.57	C++ Source	1 KB
Project1.res	13/06/2022 23.35	Compiled Resource Script	113 KB
Project1PCH1.h	13/06/2022 20.00	C/C++ Header	1 KB
Quest.cpp	13/06/2022 23.37	C++ Source	20 KB
Quest.fmx	13/06/2022 23.38	FireMonkey Form	273 KB
Quest.h	13/06/2022 23.34	C/C++ Header	4 KB
Quest.vlb	13/06/2022 23.38	VLB File	2 KB
README.md	13/06/2022 23.47	MD File	1 KB
Robot.cpp	13/06/2022 20.00	C++ Source	1 KB
Robot.h	13/06/2022 20.00	C/C++ Header	1 KB
Win.cpp	13/06/2022 22.49	C++ Source	1 KB
Win.fmx	13/06/2022 22.49	FireMonkey Form	1 KB
Win.h	13/06/2022 22.49	C/C++ Header	1 KB

Step 2: Open the 'Debug' Folder



Name	Date modified	Type	Size
Debug	13/06/2022 23.38	File folder	

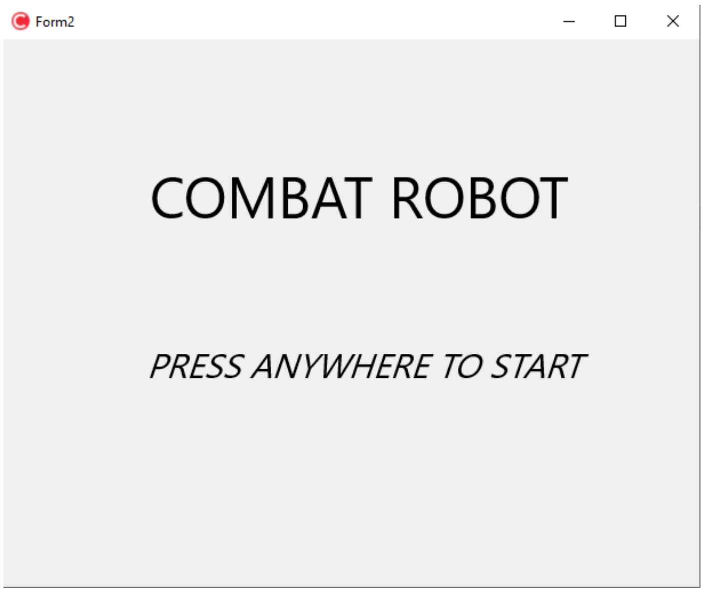
Step 3: Open Project1.exe

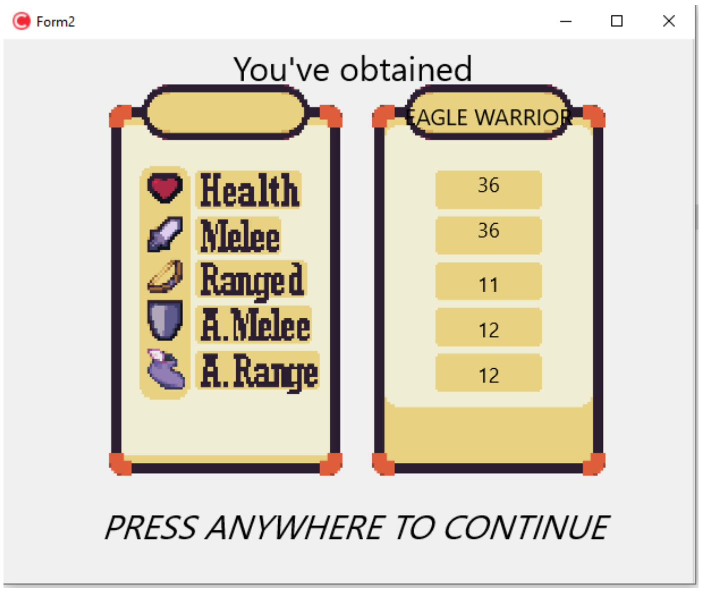


Name	Date modified	Type	Size
Account.obj	13/06/2022 20.00	3D Object	989 KB
cards.txt	13/06/2022 23.38	Text Document	2 KB
CharacterForm.obj	13/06/2022 20.00	3D Object	566 KB
Classes.obj	13/06/2022 20.00	3D Object	41 KB
GameOver.obj	13/06/2022 20.00	3D Object	538 KB
LoginForm.obj	13/06/2022 20.00	3D Object	978 KB
MainForm.obj	13/06/2022 20.00	3D Object	628 KB
Menu.obj	13/06/2022 20.00	3D Object	538 KB
NumberWizard.obj	13/06/2022 20.00	3D Object	631 KB
Project1.exe	13/06/2022 23.38	Application	600 KB
Project1.ilc	13/06/2022 23.38	ILC File	1.472 KB
Project1.ild	13/06/2022 23.38	ILD File	192 KB
Project1.ilf	13/06/2022 23.38	ILF File	5.632 KB
Project1.ils	13/06/2022 23.38	ILS File	18.560 KB
Project1.map	13/06/2022 23.38	Linker Address Map	1 KB
Project1.obj	13/06/2022 23.09	3D Object	525 KB
Project1.pdi	13/06/2022 23.38	PDI File	1 KB
Project1.tds	13/06/2022 23.38	TDS File	13.248 KB
Project1PCH1.pch	13/06/2022 20.00	Precompiled Header File	30.297 KB

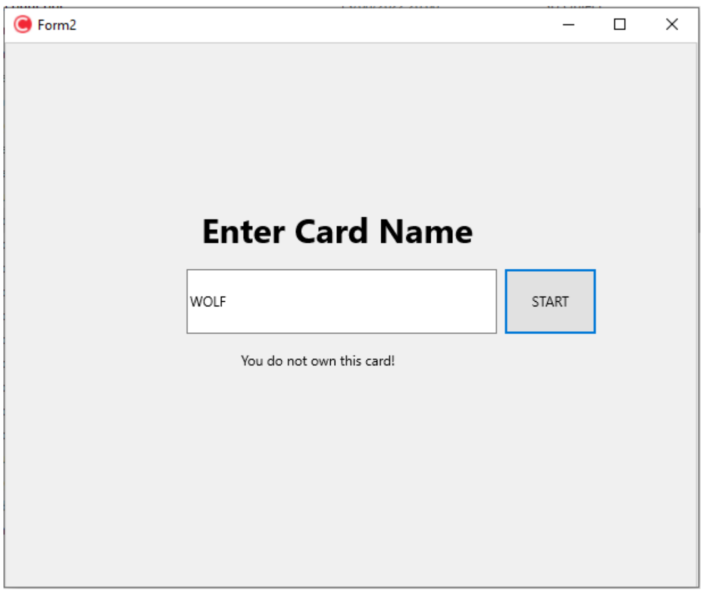
Execution of The Program

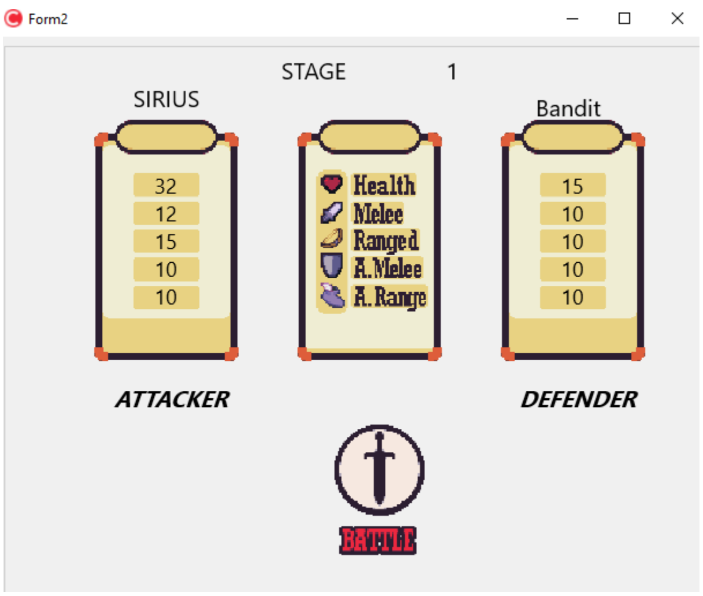
Step 1: Receive a random robot card.



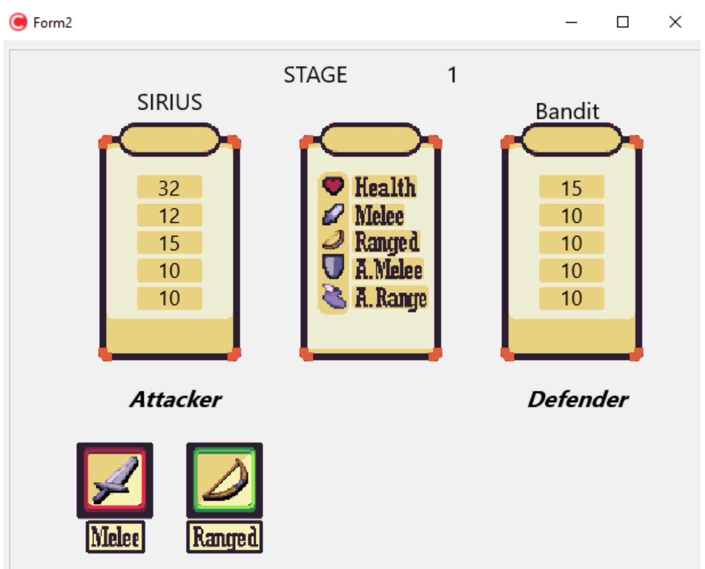


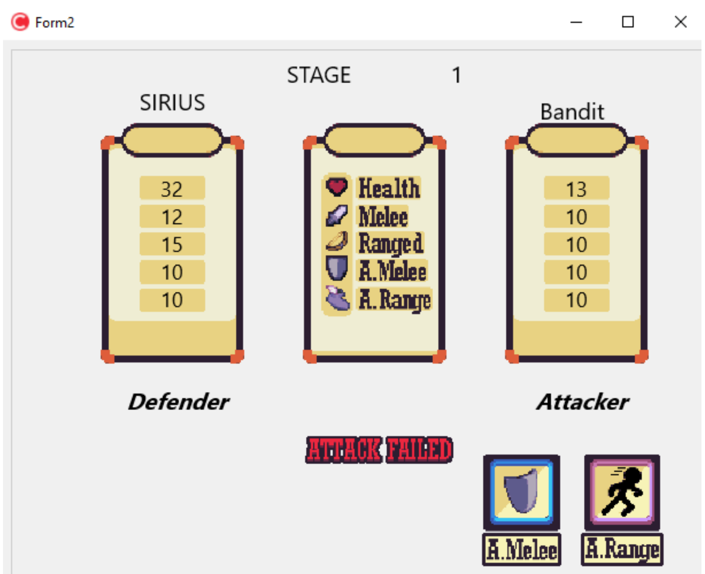
Step 2: Enter owned Card name.



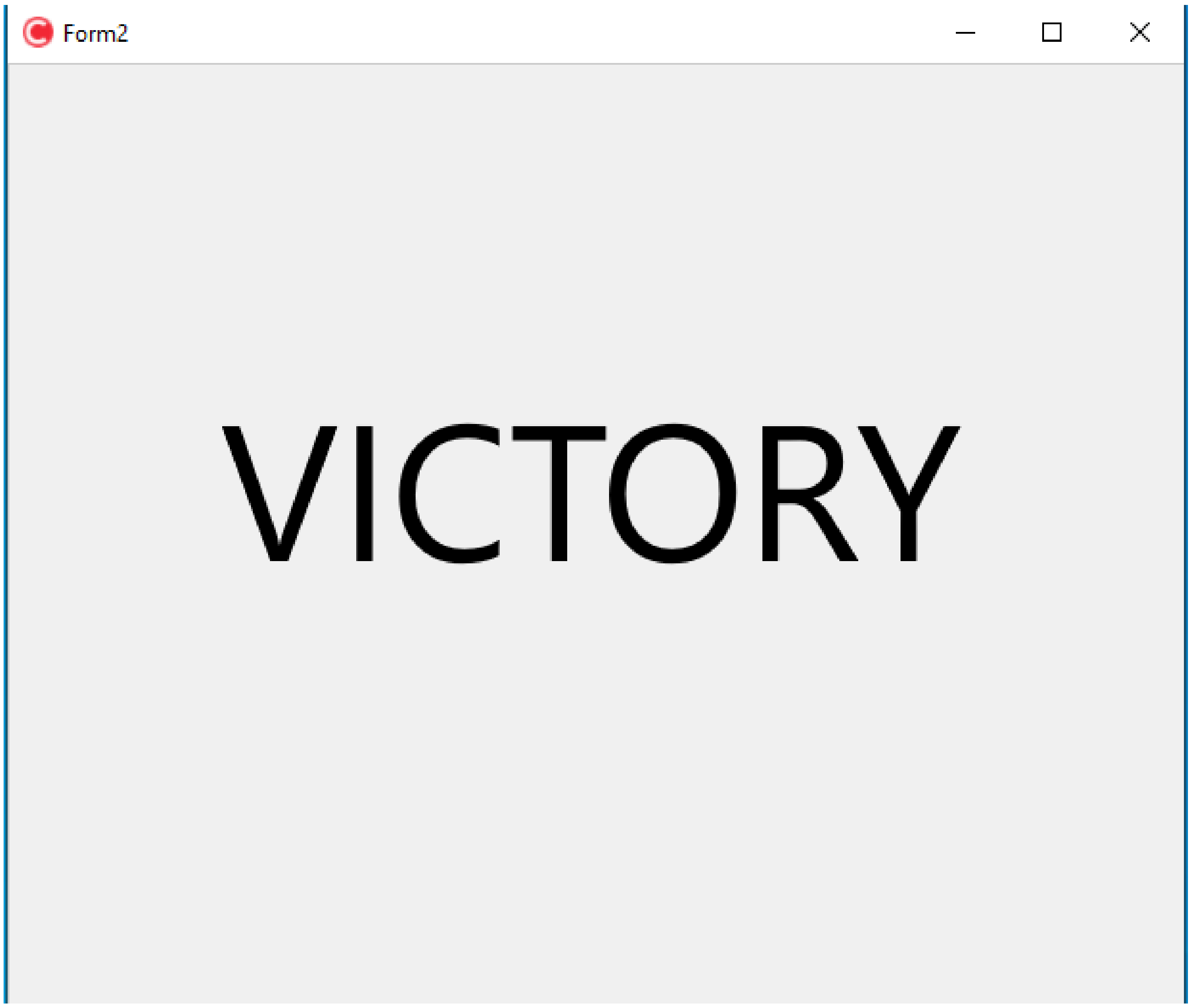


Step 3: Try to win in a combat system that depends on RNG.





Step 4: Either win or lose.



Form2

VICTORY

Resources

https://www.youtube.com/watch?v=nxHnnQToy5o&list=PL43pGnjiVwgQakzRxpt2amqN9f7-tRtc_&index=3
https://www.youtube.com/watch?v=EGCuStJyuVE&list=PL43pGnjiVwgQakzRxpt2amqN9f7-tRtc_&index=2
https://www.youtube.com/watch?v=UJI_SdxAtzk
https://thispointer.com/how-check-if-a-given-key-exists-in-a-map-c/