Cartelinars
C++ Game

C++ -ohjelmoinnin perusteet kurssin harjoitustyö
Final project for Basics of C++ Programming -course

```
                      Nhhm  N
                   Nyo++ +//ooyN
                      mys++///:oyN
                      ho////:::/:--y
                      y//o+/:::+s+/+h
                    / d / :::++::::://shh
               /soo + ::::::::::::: / d
           so / --:://:::/::::::::-d
         Nho//::++o++//::::/:::--
      Ny+///::/+++o+//:::::::-:/:-/
   ms / :: :///::/oo+/:------:----+-h
Ns//+///:+/:yo::////:+//:::::+ym
             s:::::.///////N
               d::::/smmy://+
               yossm    Nhhhm
```

Marko J. Metsola

2020

Contents
Sisällys

	Description	*Page*
	In English	
I.	Introduction	3
II.	Basics	4
III.	How	5
	Suomeksi	
I.	Johdanto	3
II.	Perusteet	6
III.	Miten	7
	Attachments / Liitteet	
I.	main.cpp	8
II.	data.cpp	10
III.	game.cpp	12
IV.	header.h	14

Introduction

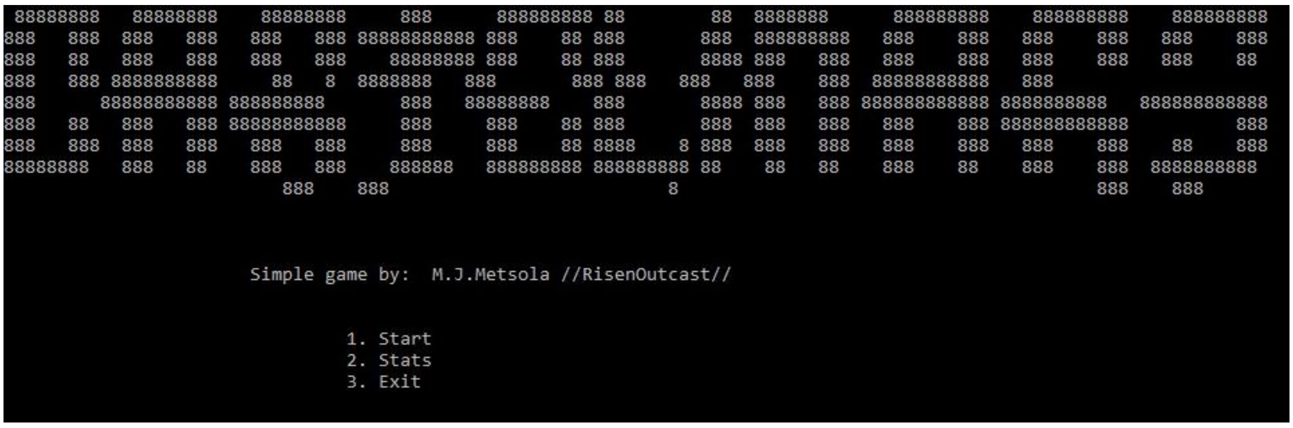
This game is made as a final project for *Basics of C++ Programming* -course

In the end I figured to make a small text-based game, after going through the requirements and the example. I hope to get a 5 for it, it fulfils all the requirements, works great and is more versatile than the examples. In addition to that it is already a good package, it is built so that you could easily add more to it and make it larger. For example, you could add a graphical user interface. Also, users' input is checked for error and asked to provide again if needed.

Johdanto

Tämä ohjelma on toteutettu C++ -ohjelmoinnin perusteet kurssia varten harjoitustyöksi.

Päädyin loppujen lopuksi tekemään pienen tekstipohjaisen pelin, kun vaatimuksia ja esimerkkejä selasin. Toivoisin ohjelmalta arvosanaa 5, se täyttää mielestäni kaikki vaatimukset, toimii hyvin ja on monipuolisempi, kuin esimerkkityöt. Sen lisäksi, että se on jo nyt hyvä paketti, se on rakennettu siten, että siihen on helppo lisätä tavaraa ja tehdä siitä isompi. Esimerkiksi lisäämällä graafisen käyttöliittymän. Käyttäjän antamat arvot myös tarkistetaan ja pyydetään uudestaan, jos niissä on ongelmia.



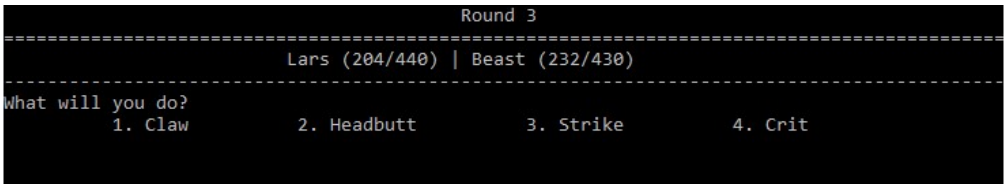
```
 88888888   88888888    88888888     888      888888888 88        88  8888888      888888888    888888888    888888888
888   888  888   888   888   888 88888888888 888    88 888       888  888888888   888    888   888    888   888    888
888   88   888   888   888   888    88888888 888    88 888       8888 888   888   888    888   888    888   888    88
888   888 8888888888     88   8  8888888   888       888 888   888   888    888  88888888888   888
888      88888888888 888888888       888   88888888    888       8888 888   888 888888888888 8888888888   888888888888
888   88   888   888 88888888888     888     888    88 888       888  888   888   888    888 88888888888           888
888   888  888   888   888   888     888     888    88 8888    8 888  888   888   888    888   888    888    88    888
88888888   888   88    888   888    888888   888888888 888888888  88   88   88    888    88    888    888  8888888888
                          888    888                          8                                       888    888


                       Simple game by:  M.J.Metsola //RisenOutcast//

                                1. Start
                                2. Stats
                                3. Exit
```

Basics

In “*Cartelinars*” you get yourself a beast, that belongs to the cartelinar -species. You can name yourself and on that point forwards you may use it in battle.

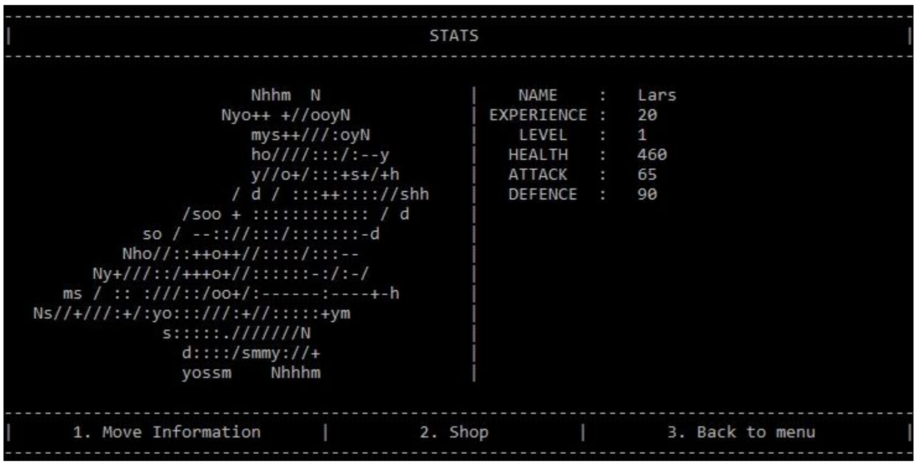
There are 4 moves, which all have different properties:

Claw	Lowers enemy’s defence at the same time.
Headbutt	Restores some health, while attacking.
Strike	Powerwolf attack, but you take recoil damage.
Crit	You have a small chance to do massive amounts of damage or very little, test your luck.



```
                                          Round 3
============================================================================================
                             Lars (204/440) | Beast (232/430)
--------------------------------------------------------------------------------------------
What will you do?
          1. Claw            2. Headbutt            3. Strike            4. Crit
```

You get experience from battles, which will power-up your beast and raise the attack, defence and health stats. You also get gold which you may spend to artificially upgrade your beast’s stats.



```
--------------------------------------------------------------------------------------------
|                                           STATS                                          |
--------------------------------------------------------------------------------------------

                         Nhhm  N                |    NAME    :   Lars
                      Nyo++ +//ooyN             | EXPERIENCE :   20
                         mys++///:oyN           |    LEVEL   :   1
                         ho////:::/:--y         |   HEALTH   :   460
                         y//o+/:::+s+/+h        |   ATTACK   :   65
                       / d / :::++:::://shh     |   DEFENCE  :   90
                  /soo + :::::::::::: / d       |
              so / --:://:::/:::::::-d          |
            Nho//::++o++//::::/:::--            |
         Ny+///::/+++o+//::::::-:/:-/           |
      ms / :: :///::/oo+/:------:----+-h        |
   Ns//+///:+/:yo:::///:+//:::::+ym             |
                s:::::.///////N                 |
                  d::::/smmy://+                |
                  yossm    Nhhhm                |

--------------------------------------------------------------------------------------------
|      1. Move Information      |         2. Shop         |         3. Back to menu        |
--------------------------------------------------------------------------------------------
```

From the Stats -page, you can check out your beast’s stats and a small reference ASCII -art.

How

The mechanics of the game are simple, let’s start with the creation of the beast. At the beginning, the program asks the user a name for the creature, after which the beast is created. Attack, Health and Defence come randomly from a list of values, that have been set beforehand. XP and Level are of course 0 at first. The player also gets its own data, which includes name and gold amount. These values are separately, to make the game easier to update. Adding an option to have multiple beasts for instance.

Before battles the game creature an enemy in a similar way and the battle happens in a while - loop, which ends when one of the creature’s health reaches 0 or less. All moves have been made into separate functions and both creatures use them, only swapping the values. Each move has its own formulas. The enemy chooses its move randomly.

With a win the players beast gets 50xp and when losing 20xp. Level up -formula is as follows: $\text{Level} = \frac{1 + \sqrt{1+8 \text{ x XP} / 50}}{2}$. The program checks if the value from the formula exceed the beast’s level and if it does, the beast level ups. Each level up raises health with 20 points, attack and defence with 10 points.

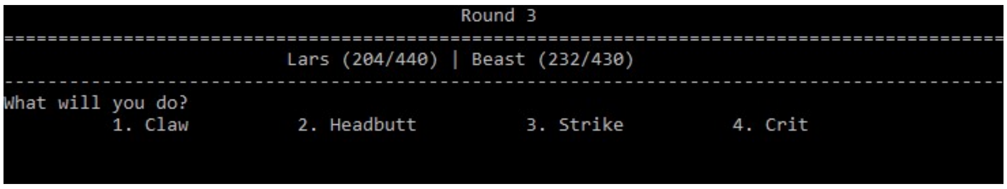
The program is divided into 3 .cpp -files and one header. Battle -related functions are found in game.cpp, menus and the start are found in main.cpp, enemy creation, player creation, shop, move list and stats are found in data.cpp. Structs and functions are collected in header.h -file and commented on from which file is the function from.

Perusteet

Cartelinars -pelissä saat itsellesi cartelinar -nimiseen lajiin kuuluvan otuksen. Saat itse nimetä otuksesi ja siitä eteenpäin voit käyttää sitä taisteluissa.

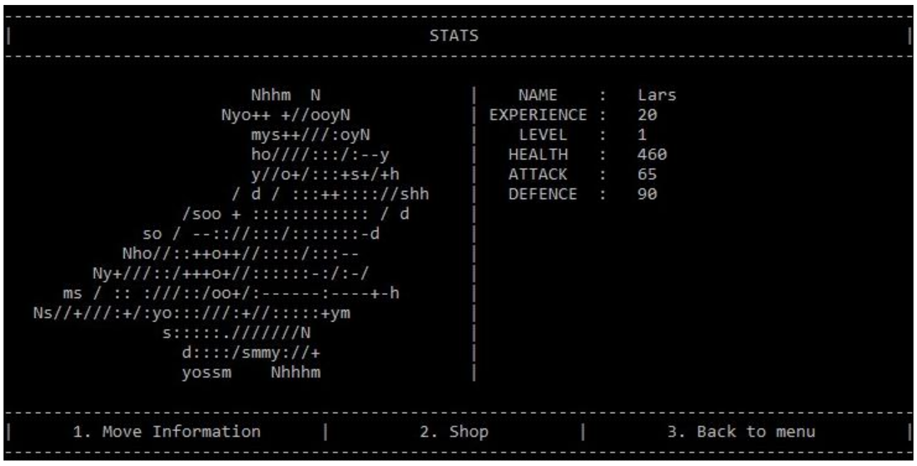
Taistelussa valittavissa on 4 eri liikettä, joilla on erilaisia ominaisuuksia:

Claw	Laskee vihollisen defence -arvoja samalla.
Headbutt	Palauttaa elämää samalla, kun tekee vahinkoa.
Strike	Voimakas isku, mutta ottaa vahinkoa itseenkin.
Crit	Arpa liike, mikäli tulee critical, isku on todella voimakas, mutta jos arpa ei osu, on isku heikoimpia.



```
                                            Round 3
==========================================================================================
                               Lars (204/440) | Beast (232/430)
------------------------------------------------------------------------------------------
What will you do?
          1. Claw          2. Headbutt          3. Strike          4. Crit
```

Taisteluista saa kokemuspisteitä (XP), jolla otuksesi saa tasoja ja attack, defence sekä health pisteet kasvavat. Sekä kultaa, jolla pystyy ostamaan kaupasta potioneita, jotka nostavat otuksesi ominaisuuksia.



```
-------------------------------------------------------------------------------------------
|                                          STATS                                          |
-------------------------------------------------------------------------------------------

                        Nhhm  N                 |    NAME     :   Lars
                     Nyo++ +//ooyN              | EXPERIENCE :   20
                        mys++///:oyN            |    LEVEL    :   1
                        ho////:::/:--y          |   HEALTH    :   460
                        y//o+/:::+s+/+h         |   ATTACK    :   65
                      / d / :::++:::://shh      |   DEFENCE   :   90
                 /soo + ::::::::::: / d         |
             so / --:://:::/:::::::-d           |
           Nho//::++o++//::::/:::--             |
        Ny+///::/+++o+//::::::-:/:-/            |
     ms / :: :///::/oo+/:------:----+-h         |
  Ns//+///:+/:yo:::///:+//:::::+ym              |
               s:::::.///////N                  |
                 d::::/smmy://+                 |
                 yossm    Nhhhm                 |

-------------------------------------------------------------------------------------------
|      1. Move Information      |          2. Shop          |         3. Back to menu      |
-------------------------------------------------------------------------------------------
```

Stats -sivulla pystyy katsomaan oman otuksen ominaisuudet ja näkee pienen ASCII -kuvan miltä otus näyttää.

Miten

Pelin mekaniikat ovat suhteellisen simppelit, aloitetaan otuksen luomisesta. Alussa ohjelma kysyy käyttäjältä otukselle nimen, jonka annettuaan otus luodaan. Attack, Health ja Defence tulevat satunnaisesti ennalta määritetyistä arvoista. XP ja Level ovat tietysti aluksi 0. Myös pelaajalle luodaan oma data, joka sisältää pelaajan nimen ja kullan määrän. Nämä arvot ovat erillään lähinnä siksi, että peli olisi helpompi muokata sellaiseksi, että yhdellä pelaajalla olisi useampi otus.

Taistelussa aluksi peli luo hyvin samalla logiikalla satunnaisen vihollisen ja itse taistelu tapahtuu while -loopissa, joka päättyy, kun jommankumman otuksen elämä menee nolliin. Liikkeet on jaettu aliohjelmiin ja ne on tehty siten, että molemmat puolet käyttävät samoja aliohjelmia, swapaten vain arvoja. Kullakin liikkeellä on erilaiset kaavat, miten ne toimivat. Vihollinen valitsee liikkeensä satunnaisesti.

Pelaajan otus saa kokemuspisteitä 50, jos tulee voitto ja 20 jos tulee häviö. Tasonnoususta vastaa kaava: $\mathrm{Taso} = \frac{1 + \sqrt{1+8 \text{ x } \mathrm{XP} / 50}}{2}$. Ohjelma tarkistaa taistelun päätteeksi, että ylittääkö kaavasta tullut taso pelaajan otuksen nykyisen tason ja jos ylittää otus nousee tasolla ja Health nousee 20 pisteellä, sekä Attack ja Defence nousevat 10 pisteellä.

Ohjelma on jaettu kolmeen .cpp -tiedostoon ja yhteen headeriin. Taisteluun liittyvät aliohjelmat löytyvät game.cpp -tiedostosta, menut ja aloitus löytyy main.cpp -tiedostosta, vihollisen luominen, kauppa, pelaajan luonti, liike -lista ja statit löytyvät data.cpp -tiedostosta. Structit löytyvät ja aliohjelmat on kätevästi kerätty header.h -tiedostoon ja perään on kommattu, kummasta tiedostosta ne löytyvät.

Attachements / Liitteet

main.cpp

```cpp
#include <iostream>
#include <Windows.h>
#include <math.h>
#include "header.h"

void letter_by_letter(std::string message) { //Esittää tekstin ruudulle kirjain kerrallaan. RPG-efekti.
    int x = 0;
    while (message[x] != '\0')
    {
        std::cout << message[x];
        Sleep(75);
        x++;
    };
    return;
}

int menu(player& currentPlayer, beast& playerBeast)
{
    int choice;

    int gameActive = 1; //Kun 1 peli pysyy käynnissä, kun 0 peli sulkeutuu.

    while (gameActive == 1) {
        system("CLS");

        std::string h = " 88888888   88888888    88888888     888      888888888 88        88  8888888      888888888
888888888    888888888 \n888   888  888   888   888   888 88888888888 888    88 888       888  888888888   888    888   888
888   888    888\n888   88   888   888   888   888    88888888 888    88 888       8888 888   888   888    888   888    888   888
88\n888   888 8888888888     88    8  8888888   888       888 888   888   888    888  88888888888   888\n888      88888888888
888888888       888   88888888    888       8888 888   888 888888888888 8888888888   888888888888\n888   88   888   888
88888888888     888     888    88 888       888  888   888   888    888 888888888888          888\n888   888  888   888   888
888     888     888    88 8888    8 888  888   888   888    888   888    888    88    888\n88888888   888   88    888   888
888888   888888888 888888888 88    88   88    888    88    888    888  8888888888\n                          888    888
8                                       888    888              \n";
        std::cout << h; //Logo pelille
        std::cout << "\n\n                       Simple game by:  M.J.Metsola //RisenOutcast//";
        std::cout << "\n\n\n                                1. Start\n                               2. Stats\n
3. Exit\n\n";
        std::cin >> choice;
        while (1) //Tarkista kayttajan antama data
        {
            if (std::cin.fail())
            {
                std::cout << "Invalid answer, enter 1,2 or 3.\n";
            }
            else
            {
                break;
            }
        }
        if (choice == 1) {
            beast enemyBeast = newEnemy(); //Luo satunnaisesti generoitu vihollinen
            battle(playerBeast, enemyBeast, currentPlayer); //aloita taistelu
        }
        else if (choice == 2)    stats(playerBeast, currentPlayer);
        else if (choice == 3)    gameActive = 0;
    }
    return 0;
}

beast newCompanion(std::string playerName) {
    int HealthChoices[15] = { 360, 400, 480, 500, 380, 470, 420, 410, 370, 490, 350, 450, 430, 440, 460 };
    int AttackChoices[8] = { 50, 55, 60, 65, 70, 75, 80, 85 };
    int DefenceChoices[8] = { 50, 55, 60, 65, 70, 75, 80, 85 };

    std::string beastName = "Cartel";
    std::string choice;

    letter_by_letter("...\n");
    Sleep(40);
    letter_by_letter("It's hatching?\n");
    letter_by_letter(playerName + ", look! It's your very own Cartelian! What should we call it?\n");
    std::cin >> beastName;

    letter_by_letter(beastName + " is an excellent name! Now go out and win some battles!\n");
    std::cout << "Enter 1 to continue.\n";
    while (choice.size() != 1 || 1) {
        std::cin >> choice;
        if (choice != "1") {
            std::cout << "Invalid answer! Enter 1 to continue!\n";
        }
        else {
            break;
        }
    }


    struct beast playerBeast;
    playerBeast.name = beastName;
    playerBeast.xp = 0;
    playerBeast.level = 0;
    playerBeast.health = HealthChoices[std::rand() % 14 + 0]; //Satunnainen elamapisteidenmaara ennalta maaritetyista vaihtoeh-
doista
    playerBeast.attack = AttackChoices[std::rand() % 7 + 0];//Satunnainen attack pisteiden maara ennalta maaritetyista vaihtoeh-
doista
    playerBeast.defence = DefenceChoices[std::rand() % 7 + 0];//Satunnainen defence pisteiden maara ennalta maaritetyista vaihto-
ehdoista

    return playerBeast;
}

int main() {
    std::string choice;
    std::string valid_choices("yYnN");

    letter_by_letter("Welcome to the world of Cartepodare!\nDo you want to hear the introduction? [Y]/[N]\n"); //Let's start by
getting you a bird.\n
    std::cin >> choice;
    while (choice.size() != 1 || valid_choices.find(choice) == std::string::npos)
    {
        std::cout << "Invalid answer, enter Y or N.\n";
        std::cin >> choice;
    }
    if (choice == "Y") {
        system("CLS");
        letter_by_letter("Cartepodare, also known as Cartelinars, are a smal species with immense fighting ability.These little
things live to fight and have a strong connection to humans. Using humans to guide them in battle, we use them for entertainment
and competition!\nGo get your own Cartelinar and win some prizes!\n");
        std::cout << "Enter 1 to continue.";
        while (choice.size() != 1 || 1) {
            std::cin >> choice;
            if (choice != "1") {
                std::cout << "Invalid answer! Enter 1 to continue!";
            }
            else {
                break;
            }
        }
    }

    system("CLS");

    letter_by_letter("What is your name?\n\n");
    std::cin >> choice;

    player Player = newPlayer(choice); //Luo pelaajan data

    beast Beast = newCompanion(choice); //Luo beastin data

    menu(Player, Beast);
}
```

data.cpp

```cpp
#include <iostream>
#include "header.h"

beast newEnemy() {
    struct beast playerBeast;

    int HealthChoices[15] = { 360, 400, 480, 500, 380, 470, 420, 410, 370, 490, 350, 450, 430, 440, 460 }; //Arvoja mista arpoa
    int AttackChoices[8] = { 50, 55, 60, 65, 70, 75, 80, 85 }; //Arvoja mista arpoa
    int DefenceChoices[8] = { 50, 55, 60, 65, 70, 75, 80, 85 }; //Arvoja mista arpoa

    playerBeast.name = "Beast"; //Tamankin voisi periaatteessa arpoa
    playerBeast.xp = 0; //Ei ole merkitysta, joten 0
    playerBeast.level = 0; //Ei ole merkitysta, joten 0
    playerBeast.health = HealthChoices[std::rand() % 14 + 0]; //Satunnainen elamapisteidenmaara ennalta maaritetyista vaihtoehdoista
    playerBeast.attack = AttackChoices[std::rand() % 7 + 0]; //Satunnainen attack pisteiden maara ennalta maaritetyista vaihtoehdoista
    playerBeast.defence = DefenceChoices[std::rand() % 7 + 0]; //Satunnainen defence pisteiden maara ennalta maaritetyista vaihtoehdoista

    return playerBeast;
}

player newPlayer(std::string userName) { //Uuden pelaajan luonti
    struct player currentPlayer;
    currentPlayer.name = userName;
    currentPlayer.gold = 0;

    return currentPlayer;
}

void shop(beast& playerBeast, player& currentPlayer) {
    std::string choice;
    int oldStat = 0;
    int active = 1; //Kun aktiivinen niin voi tehdä useamman oston.


    system("CLS"); //Tyhjaa ruutu
    std::cout << "--------------------------------------------------------------------------------------------\n";
    std::cout << "|                                          SHOP                                            |\n";
    std::cout << "--------------------------------------------------------------------------------------------\n";
    std::cout << "|     1. Attack Potion      |       2. Defence Potion      |        3. Health Potion       |\n";
    std::cout << "|      Slight increase      |        Slight increase       |         Slight increase       |\n";
    std::cout << "|            50g            |              50g             |               50g             |\n";
    std::cout << "--------------------------------------------------------------------------------------------\n\n";
    std::cout << "\n--------------------------------------------------------------------------------------------\n";
    std::cout << "|                                      4. Back to Menu                                     |";
    std::cout << "\n--------------------------------------------------------------------------------------------\n\n";

    while (active == 1) {
        std::cout << "You currently hold " << currentPlayer.gold << " in your wallet.\n";
        std::cin >> choice;
        if (choice == "4") {
            active = 0;
            return;
        }
        if (currentPlayer.gold > 49) {
            if (choice == "1")
            {
                oldStat = playerBeast.attack; //Ota talteen vanha arvo, jotta se voidaan esittaa pelaajalle.
                playerBeast.attack += 50;
                std::cout << playerBeast.name << " attack power has now icreased from " << oldStat << " to " << playerBeast.attack << "\n";
                currentPlayer.gold -= 50;
            }
            if (choice == "2")
            {
                oldStat = playerBeast.defence; //Ota talteen vanha arvo, jotta se voidaan esittaa pelaajalle.
                playerBeast.defence += 50;
                std::cout << playerBeast.name << " defence has now icreased from " << oldStat << " to " << playerBeast.defence << "\n";
                currentPlayer.gold -= 50;
            }
            if (choice == "3")
            {
                oldStat = playerBeast.health; //Ota talteen vanha arvo, jotta se voidaan esittaa pelaajalle.
                playerBeast.health += 50;
                std::cout << playerBeast.name << " health has now icreased from " << oldStat << " to " << playerBeast.health << "\n";
                currentPlayer.gold -= 50;
            }
        }
        else {
            std::cout << currentPlayer.name << ", you do not have enough gold.\n";
        }
    }
}
void moves() { //Tietoa liikkeista.
    int choice;
    int oldStat = 0;
    int active = 1; //Kun aktiivinen niin voi tehdä useamman oston.


    system("CLS"); //Tyhjaa ruutu
    std::cout << "======================================================================================================================\n";
    std::cout << "|                                                     Move List                                                      |\n";
    std::cout << "======================================================================================================================\n";
    std::cout << "|                                                       Claw                                                         |\n";
    std::cout << "|        Beast's claws are like strong swords, striking with them they can even lower enemy´s defence slightly!     |\n";
    std::cout << "---------------------------------------------------------------------------------------------------------------------\n";
    std::cout << "|                                                     Headbutt                                                      |\n";
    std::cout << "|                       Beast headbutts the enemy so hard that it gets some health points back!                     |\n";
    std::cout << "---------------------------------------------------------------------------------------------------------------------\n";
    std::cout << "|                                                      Strike                                                       |\n";
    std::cout << "|        Beast goes berserk and deals massive damage, but without realising it, it tooks some recoil damage!        |\n";
    std::cout << "---------------------------------------------------------------------------------------------------------------------\n";
    std::cout << "|                                                       Crit                                                        |\n";
    std::cout << "|             Theres a chance this move makes massive amounts of damge or it deals little to nothing!               |\n";
    std::cout << "=====================================================================================================================\n";
    std::cout << "|                                                 1. Back to Menu                                                   |\n";
    std::cout << "=====================================================================================================================\n\n";
    std::cin >> choice;
    while (1) //Tarkista kayttajan antama data
    {
        if (std::cin.fail())
        {
            std::cout << "Invalid answer, enter 1.\n";
        }
        else
        {
            break;
        }
    }
    if (choice == 1)          return;
}

void stats(beast& playerBeast, player& currentPlayer) { //Hahmon ja pelaajan tilastot.
    int choice;
    int active = 1; //Kun aktiivinen niin statit päivittyvät, kun tulet takaisin toisesta esim. shopista.
    while (active == 1) {
        system("CLS");
        std::cout << "--------------------------------------------------------------------------------------------\n";
        std::cout << "|                                          STATS                                           |\n";
        std::cout << "--------------------------------------------------------------------------------------------\n\n";

        std::cout << "                         Nhhm  N               " << "|    NAME    :   " << playerBeast.name << "\n";
        std::cout << "                      Nyo++ +//ooyN            " << "| EXPERIENCE :   " << playerBeast.xp << "\n";
        std::cout << "                         mys++///:oyN          " << "|    LEVEL   :   " << playerBeast.level << "\n";
        std::cout << "                         ho////:::/:--y        " << "|   HEALTH   :   " << playerBeast.health << "\n";
        std::cout << "                         y//o+/:::+s+/+h       " << "|   ATTACK   :   " << playerBeast.attack << "\n";
        std::cout << "                       / d / :::++:::://shh    " << "|   DEFENCE  :   " << playerBeast.defence << "\n";
        std::cout << "                  /soo + :::::::::::: / d      " << "|\n";
        std::cout << "              so / --:://:::/:::::::-d         " << "|\n";
        std::cout << "            Nho//::++o++//::::/:::--           " << "|\n";
        std::cout << "         Ny+///::/+++o+//::::::-:/:-/          " << "|\n";
        std::cout << "      ms / :: :///::/oo+/:------:----+-h       " << "|\n";
        std::cout << "   Ns//+///:+/:yo::///:+//::::::+ym            " << "|\n";
        std::cout << "                s:::::.///////N                " << "|\n";
        std::cout << "                  d::::/smmy://+               " << "|\n";
        std::cout << "                  yossm    Nhhhm               " << "|\n";
        std::cout << "\n--------------------------------------------------------------------------------------------\n";

        std::cout << "|      1. Move Information      |         2. Shop         |        3. Back to menu         |";
        std::cout << "\n--------------------------------------------------------------------------------------------\n\n";
        std::cin >> choice;

        while (1) //Tarkista kayttajan antama data
        {
            if (std::cin.fail())
            {
                std::cout << "Invalid answer, enter 1,2 or 3.\n";
            }
            else
            {
                break;
            }
        }
        if (choice == 1)          moves();
        else if (choice == 2)     shop(playerBeast, currentPlayer);
        else if (choice == 3)     return;
    }
}
```

game.cpp

```cpp
#include <iostream>
#include <Windows.h>
#include <math.h>
#include "header.h"

void move1(beast& beast_attacker, beast& beast_defender) { //Claw   -Heikompi isku, mutta laskee vihollisen defence pisteita.
    int EnemyDefenceDec = beast_defender.defence * 0.05; //Laske 5% beastin defence arvosta
    beast_defender.defence -= EnemyDefenceDec; //vahenna se beastin defencesta
    int damage = beast_attacker.attack * beast_attacker.attack / (beast_attacker.attack + beast_defender.defence); //Laske 
vahinko
    beast_defender.health -= damage; //Vahenna se vihollisen elamapisteista
    std::cout << beast_attacker.name << " uses Claw! " << beast_defender.name << " defence fell!\n" << beast_attacker.name << " 
deals " << damage << " damage! " << beast_defender.name << " has " << beast_defender.health << " hp left!\n\n";
    return;
}

void move2(beast& beast_attacker, beast& beast_defender) { //Headbutt   -Heikompi isku, mutta antaa samalla vahan elamapisteita 
takaisin
    int healAmount = beast_attacker.health * 0.05; //Laske 5% beastin elamapisteista
    beast_attacker.health += healAmount; //Lisaa laskettu 5% beastin elamapisteisiin
    int damage = beast_attacker.attack * beast_attacker.attack / (beast_attacker.attack + beast_defender.defence); //Laske va-
hinko
    beast_defender.health -= damage; //Vahenna se vihollisen elamapisteista
    std::cout << beast_attacker.name << " uses Headbutt! " << beast_attacker.name << " deals " << damage << " damage! \n" << 
beast_defender.name << " has " << beast_defender.health << " hp left!\n\n";
    return;
}

void move3(beast& beast_attacker, beast& beast_defender) { //Strike -Voimakkaampi isku, mutta tekee vahinkoa myos itselle.
    int selfharm = beast_attacker.defence * 0.1; //Laske 10% beastin hyokkausvoimasta
    beast_attacker.health -= selfharm;  //Vahenna laskettu maara omista elamapisteista
    int damage = beast_attacker.attack * 1.7; //Vahinko on 170% omista attack power pisteista
    beast_defender.health -= damage; //Vahenna se vihollisen elamapisteista
    std::cout << beast_attacker.name << " uses Strike! " << beast_defender.name << " defence fell!\n" << beast_attacker.name << " 
deals " << damage << " damage! " << beast_defender.name << " has " << beast_defender.health << " hp left!\n\n";
    return;
}

void move4(beast& beast_attacker, beast& beast_defender) { //Crit  -Arpa isku, jos saat "critin" on isku 150% voimakkaampi tai 
jos et, on se vain 50% suuruinen.

    int damage = beast_attacker.attack * beast_attacker.attack / (beast_attacker.attack + beast_defender.defence);//Laske vahinko
    int numero = std::rand() % 6 + 1; //Satunnainen numero valilta 1 - 6
    numero == 3 ? damage = damage * 2.5 : damage = damage * 0.5; // Jos arvottu numero oli 3 isku on 250% jos ei 50%
    beast_defender.health -= damage; //Vahenna vahinko vihollisen elamapisteista
    numero == 3 ?
        std::cout << beast_attacker.name << " uses Crit! " << beast_attacker.name << " deals " << damage << " damage! " << 
"\nCRITICAL Strike! " << beast_defender.name << " has " << beast_defender.health << " hp left!\n\n" :
        std::cout << beast_attacker.name << " uses Crit! " << beast_attacker.name << " deals " << damage << " damage! " << "\nIt 
didn't go as planned. " << beast_defender.name << " has " << beast_defender.health << " hp left!\n\n";
    return;
}

void add_experience(beast& playerBeast, player& currentPlayer, int xp_amount, int gold_amount) {
    playerBeast.xp += xp_amount;
    currentPlayer.gold += gold_amount;
    std::string choice;
    int level_end = (1 + sqrt(1 + 8 * playerBeast.xp / 50)) / 2; //Laske mikä leveli experiencesta tulee.

    if (playerBeast.level < level_end) { //Vertaa laskettua leveliä beastin leveliin
        playerBeast.attack += 10;
        playerBeast.defence += 10;
        playerBeast.health += 20;
        playerBeast.level = level_end;
        std::cout << "Your " << playerBeast.name << " has grown to level " << level_end << "! \n";
        std::cout << "Attack increased to " << playerBeast.attack << "! Defence increased to " << playerBeast.defence << "! 
Health increased to " << playerBeast.health << "! \n";

        std::cout << "Enter 1 to continue.\n";
        while (choice.size() != 1 || 1) { //Tarkista kayttajan antama data.
            std::cin >> choice;
            if (choice != "1") {
                std::cout << "Invalid answer! Enter 1 to continue!\n";
            }
            else {
                break;
            }
        }
    }
    std::cout << "You got " << gold_amount << " gold!\n";

    std::cout << level_end;
    return;
}

int battle(beast& playerBeast, beast& enemyBeast, player& currentPlayer) {
    int round = 0;

    beast player = playerBeast; //Ota beasteista kopiot, jotta alkuperaiset tiedot eivat muutu.
    beast enemy = enemyBeast;

    int player_currenthealth = playerBeast.health; //Ota talteen pelaajan max elamapisteet, jotta niitä voidaan vahentaa
    int enemy_currenthealth = enemy.health; //Ota talteen vihollisen max elamapisteet, jotta niitä voidaan vahentaa

    int playerInput = 0; //Pelaajan tekema valinta.

    int winner = 0; //Kaytetaan kun matsi on ohi.


    while (1) {
        system("CLS");
        round += 1;
        std::cout << "                                          Round " << round << "\n";
        std::cout << "===========================================================================================\n";
        std::cout << "                          " << player.name << " (" << player.health << "/" << playerBeast.health << ")" << 
" | " << enemyBeast.name << " (" << enemy.health << "/" << enemyBeast.health << ")" << "\n";
        std::cout << "-------------------------------------------------------------------------------------------\n";

        //Pelaajan vuoro
        std::cout << "What will you do?\n          1. Claw          2. Headbutt          3. Strike          4. Crit\n";
        std::cin >> playerInput;

        if (playerInput == 1)         move1(player, enemy);
        else if (playerInput == 2)    move2(player, enemy);
        else if (playerInput == 3)    move3(player, enemy);
        else if (playerInput == 4)    move4(player, enemy);
        else    std::cout << "Invalid answer! " << player.name << " didn't know what to do and did nothing! \n"; //Tarkoituksella 
beast sekaantuu, jos kayttaja ei osaa antaa oikeaa komentoa.

        if (enemy.health < 1) { //Jos vihollinen kuoli, paata peli
            winner = 1;
            break;
        }

        Sleep(4500); //Annetaan pelaajalle aikaa lukea tapahtumat.
        //Vihollisen vuoro
        int numero = std::rand() % 4 + 1; //Satunnainen numero valilta 1 - 4, joka maarittaa mita iskua vihollinen kayttaa.
        if (numero == 1)         move1(enemy, player);
        else if (numero == 2)    move2(enemy, player);
        else if (numero == 3)    move3(enemy, player);
        else if (numero == 4)    move4(enemy, player);

        Sleep(4500);

        if (player.health < 1) { //Jos pelaaja kuoli, paata peli
            break;
        }
    }

    if (winner == 1) {
        std::cout << "You have won " << currentPlayer.name << "! \n";
        Sleep(3500);
        add_experience(playerBeast, currentPlayer, 50, 10); //Pelaajan palkinto 50xp ja 10g
    }
    else {
        std::cout << "You have lost " << currentPlayer.name << "... \n";
        Sleep(3500);
        add_experience(playerBeast, currentPlayer, 20, 0); //Pelaajan palkinto 20xp ja 0g
    }

    return 0;
}
```

header.h

```
#pragma once

struct beast { //Pelaajan data ja beastin data erikseen, jotta tulevaisuudessa voi olla useampi beast yhdella pelaajalla.
    std::string name;
    int xp;
    int level;
    int health;
    int attack;
    int defence;
};

struct player { //Pelaajan data ja beastin data erikseen, jotta tulevaisuudessa voi olla useampi beast yhdella pelaajalla.
    std::string name;
    int gold;
};

beast newEnemy(); //data.cpp
player newPlayer(std::string userName); //data.cpp

void shop(beast& playerBeast, player& currentPlayer); //data.cpp
void moves(); //data.cpp
void stats(beast& playerBeast, player& currentPlayer); //data.cpp

void move1(beast& beast_attacker, beast& beast_defender); //game.cpp
void move2(beast& beast_attacker, beast& beast_defender); //game.cpp
void move3(beast& beast_attacker, beast& beast_defender); //game.cpp
void move4(beast& beast_attacker, beast& beast_defender); //game.cpp
void add_experience(beast& playerBeast, player& currentPlayer, int xp_amount, int gold_amount); //game.cpp
int battle(beast& playerBeast, beast& enemyBeast, player& currentPlayer); //game.cpp
```