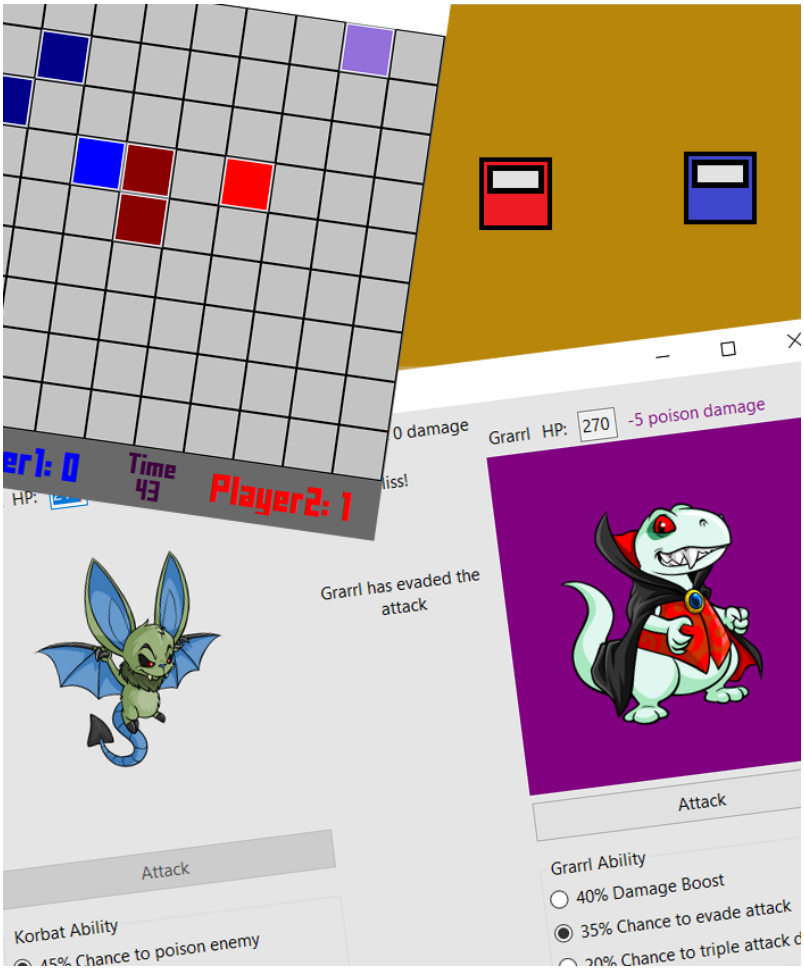
Startsemester oriëntatieverslag
Leeruitkomst Software



Studentnaam: Kaan G
Studentnummer: 457632
Klas: PD02
Vakdocent: Coen Crombach

Versie: 1.0
Datum: 22-5-2021

Inhoudsopgave

1 Inleiding

1.1 Aanleiding

Ik schrijf dit verslag om aan te tonen dat ik de verdiepende stof van software onder de knie heb.

1.2 Onderwerp

Het verslag gaat over hoe ik alle leerdoelen heb aangetoond.

1.3 Leeswijzer

In dit verslag ga je telkens een leerdoel zien. Vervolgens laat ik met behulp van mijn programma's zien hoe ik de leerdoelen heb aangetoond. Zoals je hierboven ziet probeer ik minder relevante dingen zo kort mogelijk te houden. Op de interessante onderwerpen gaan we meer op detail in.

2 Introductie

Wie ben ik?

Mijn naam is Kaan G en ik ben al 18 jaar oud. Voordat ik naar semester 1 ICT was gekomen, heb ik nog semester 2 media gevolgd. Dit bleek echt niks voor mij te zijn. Dit keer heb ik software gekozen. Bij software heb ik veel meer plezier dan bij media, dus het was zeker een goede keuze. Mijn interesses liggen vooral bij sport en games. Het lijkt me zeker leuk om later games te developen. Maar wie weet verandert dat in de toekomst.

Wat wilde ik graag leren in het startsemester?

Binnen het startsemester wilde ik graag leren hoe je met meerdere mensen aan één softwareprogramma werkt. Bij dit proces hoort dan ook het maken van crc-kaarten/class diagram. Verder wilde ik ook leren hoe je codeert met classes en objects, hoe je properties en constructors gebruikt en wat encapsulation is.

3 Aantonen leerdoelen

3.1 Proof of concepts

Hier laat ik zien hoe ik de volgende onderwerpen heb toegepast in mijn programmas.

3.1.1 objects/ classes

Mijn allereerste project waar ik classes heb gebruikt is in de BattleSim opdracht(Dit is een opdracht waar 2 karakters tegen elkaar vechten door middel van knoppen). Deze opdracht heb ik gemaakt toen ik nog bij media semester 2 zat. Een vriend van me zei dat het een leuke opdracht was daarom is dit de eerste opdracht die ik met classes heb gemaakt.

```
4 references
class Fighter
{
    // create random object (rnd)
    Random rnd = new Random();

    // create field
    public int health;

    // normal attack
    6 references
    public int Attack()
    {
        // generate random number (0-30)
        int attack = rnd.Next(0, 31);

        return attack;
    }

    1 reference
    public int Attack_NeverMiss()
    {
        // generate random number (0-30)
        int attack = rnd.Next(1, 31);

        return attack;
    }
```

Ik heb langs de form class een fighter class gemaakt. Hier gebeuren de dingen die met de fighters te maken hebben.

Wat je in deze class ziet is ten eerste dat ik een random object aanmaak, deze heb ik gebruikt om de attack power te randomizen. Verder maak ik ook nog een field met de naam 'health'. Deze field is public omdat ik later in de form class de waardes wilde veranderen. Verder zie je nog methods voor een paar attacks. Ben hier niet zo efficient te werk gegaan, had er ook geen crc kaarten bij gemaakt.

Hier een recenter voorbeeld

In onze design challenge hebben we het project GeoBlastr gemaakt, dit is een game waarin kinderen de locaties van continenten kunnen leren. Binnen dit project heb ik samen met Ruben gecodeerd. Voordat we waren begonnen met coderen hadden we crc cards gemaakt. Die zien er als volgt uit.

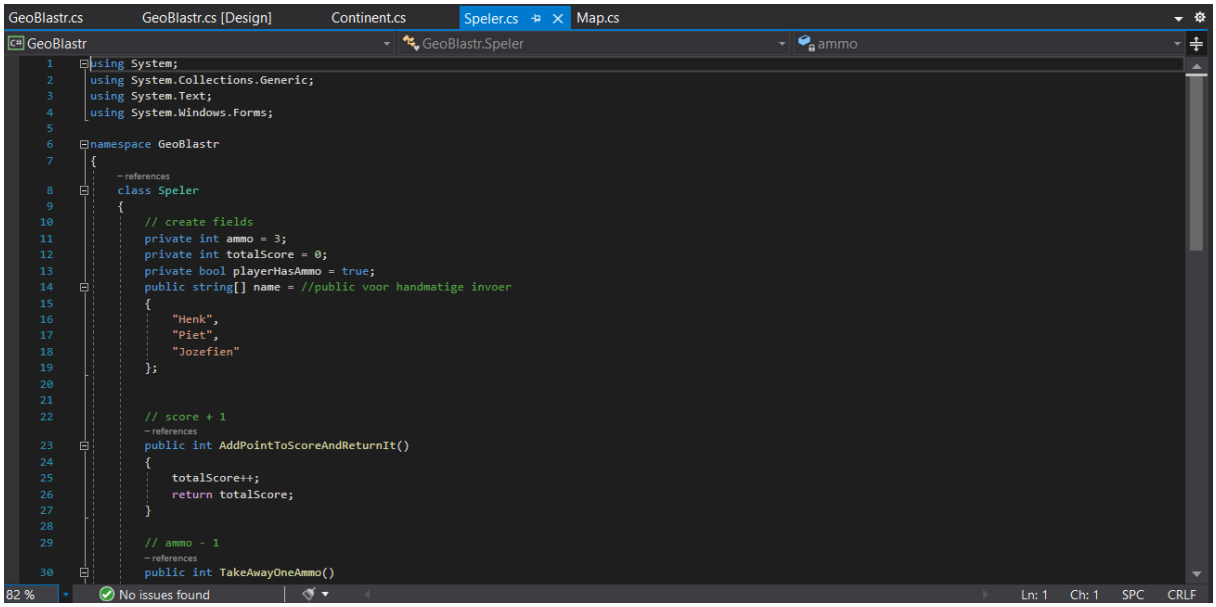
Speler	
1	Beschikbare ammo
2	Totale score
3	Positie op de kaart
4	Naam Speler

Continent	
1	Speler wie het raad
2	Is het geraden?
3	Naam van het continent
1	Speler
2	Map

Map	
1	Zet de continenten op de goede plek.
2	Buttons aanmaken
1	Continent
2	Game

GeoBlastr	
1	Begin van het spel
2	Einde van het spel
1	Map
2	Speler

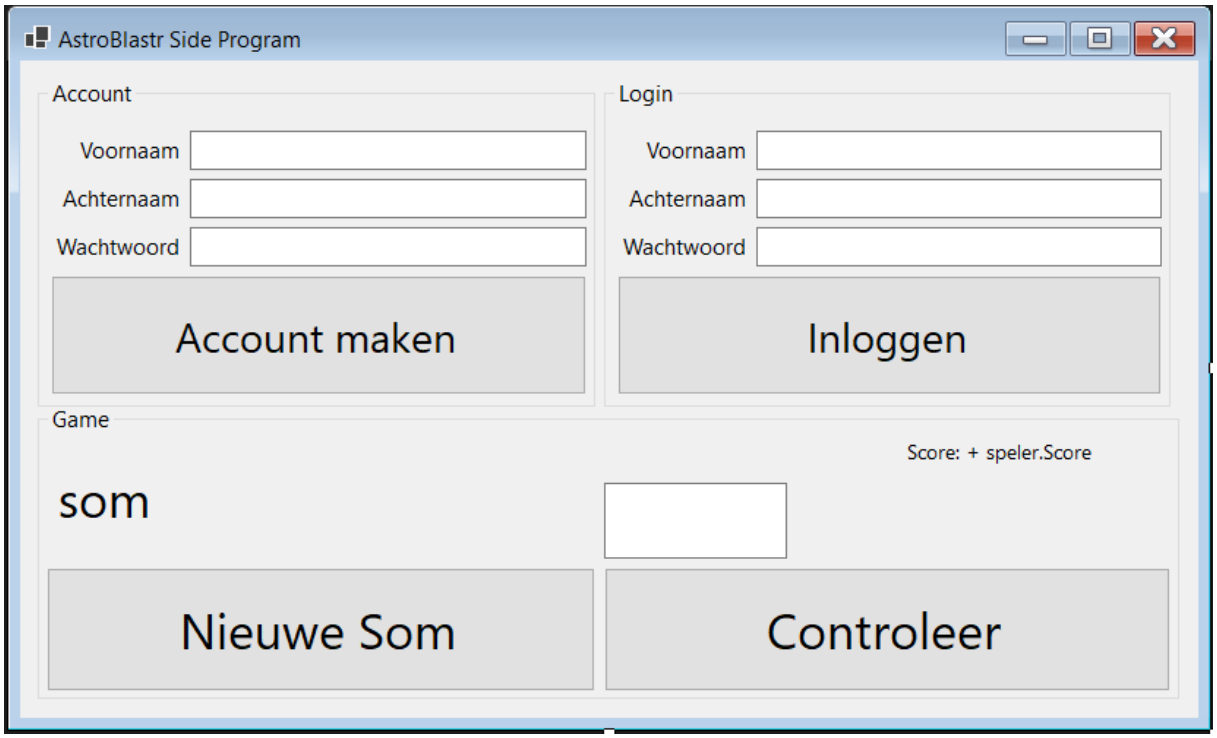
We hebben als feedback gekregen dat we beter moesten opzoeken wat het verschil was tussen class diagram en crc kaarten. Jammer genoeg hadden we veel te laat feedback gevraagd dus kwamen we daar niet meer aan toe. Toch was dit erg handig voor ons, zo wisten we welke classes we nog moesten maken en wat er zo ongeveer in moest.



Hier is nog een afbeelding uit het programma, weet niet echt wat ik nog moet laten zien.

3.1.2 Constructors

Ik heb ooit met een constructor gewerkt. Ik en ruben werken in de talk to me aan een game genaamd AstroBlastr. Het programma staat op Ruben's pc. Daarom heb ik nog een klein programmatje waar ik mijn classes maak en dingen test. Dat programma ziet er momenteel zo uit.



We hebben de volgende classes in ons programma: Speler, Som en Account.

```
3 references
class Account
{
    // Fields
    private string firstName;
    private string lastName;
    private string password;

    // Properties
    1 reference
    public string FirstName { get { return firstName; } }
    1 reference
    public string LastName { get { return lastName; } }
    1 reference
    public string Password { get { return password; } }

    // Constructors
    1 reference
    public Account(string firstName, string lastName, string password)
    {
        this.firstName = firstName;
        this.lastName = lastName;
        this.password = password;
    }
}
```

Elke keer dat we een Account object aanmaken in de form class dan moet je eerst drie textboxen invullen. De drie gegevens in de textboxen worden doorgegeven aan de constructor. Vervolgens worden de waarden gegeven aan de private fields. Het is gelijk dus ook een voorbeeld van encapsulation. Je ziet ook dat ik properties heb staan in de class waar alleen get staat. Hierdoor kun je de waarden uit de private string halen in een andere class, maar de waarden kunnen niet verandert worden.

```
// maak account object aan
private Account account;


1 reference
private void BTN_createAccount_Click(object sender, EventArgs e)
{
    // check ofdat de vakjes ingevuld zijn en of dat het wachtwoord wel minstens 4 tekens heeft
    if (TB_firstName.Text.Length > 0 && TB_lastName.Text.Length > 0 && TB_password.Text.Length > 3)
    {
        // constructor toepassen
        account = new Account(TB_firstName.Text, TB_lastName.Text, TB_password.Text);

        MessageBox.Show("Account aangemaakt");
        GB_account.Enabled = false;
        GB_Login.Enabled = true;
    }
    else if (TB_firstName.Text.Length > 0 && TB_lastName.Text.Length > 0 && TB_password.Text.Length >= 0)
    {
        MessageBox.Show("Wachtwoord moet minimaal 4 tekens lang zijn");
    }
    else
    {
        MessageBox.Show("Voer geldige gegevens in");
    }
}
```

3.1.3 encapsulation: private fields, get/set- methods en/of properties

Kijk hierboven

3.1.4 method/constructor overloading

Method overloading zie je terug in het programma "RPG".

```
public string AmogusYellow(int InterActionsWithYellow, int InteractionsWithPink)
{
    string text;
    if (InterActionsWithYellow == 0)
    {
        text = "Hey, watch out for pink. he's been following me for a while. He's pretty sus";
    }
    else if (InteractionsWithPink > 0)
    {
        text = "Wait, he said bring me the knife? I'm starting a meet, lets vote him out";
    }
    else
    {
        text = "Pink is not even doing tasks, he's so fricking sus";
    }
    return text;
}

public string AmogusYellow()
{
    string text = "Ok bro, I understand, we should play safe";
    return text;
}

public string AmogusYellow(bool knife)
{
    string text = "WHY DO YOU HAVE A KNIFE!";
    return text;
}
```

Hier een stukje uit de "Dialogue" class. Je krijgt verschillende dialogen gebasseerd op wat je als parameter meegeeft in de form. Stel je hebt op het moment een mes dan krijg je een ander dialoog dan als je geen mes hebt.

3.1.5 class diagrams/ relations (tussen classes)/ multiplicity.

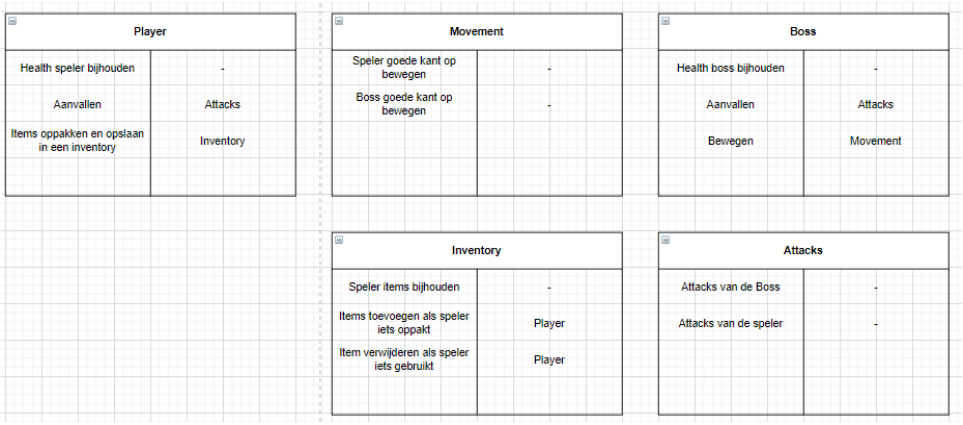
Dialogue	
Met andere karakters praten	-
Ander dialoog als je bepaalde items hebt	Inventory

Inventory	
Bijhouden welke items je wel en niet hebt	-

Forms	
Rond bewegen	Movement
Dingen oppakken	-
Niet door dingen heen bewegen	-
Kunnen praten met mensen	Dialogue

Movement	
Speler juiste directie doorgeven	-
Speler juiste kant op bewegen	-

Hier zie je de crc cards van het programma "RPG". Als extraatje noteer ik per taak met welke class die samen werkt. Hier nog een voorbeeld van een programma die ik wilde gaan maken maar nog niet heb gemaakt.



Beide bestanden kun je terug vinden bij "Bijlages" in de zipfile.

3.1.6 methods die als parameter/returnvalue-type ook eigen gemaakte typen kunnen hebben

In het volgende voorbeeld zie je dat ik in de form class een picturebox aanmaak en dat ik deze picturebox heen en weer stuur naar de Block class.

```
3 references
public partial class FRM_sirted : Form
{
    // create objects from other classes to work together
    Blocks blocks = new Blocks();
    Collisions collisions = new Collisions();
    PictureBox block = new PictureBox();
    Point[] existingBlocks = new Point[72];

    1 reference
    public FRM_sirted()
    {
        InitializeComponent();
    }

    // these numbers will modify the location of the block
    int xLocationModifier = 0;
    int yLocationModifier = 0;
```

Deze picturebox genaamd 'block' heeft dan nog geen values.

In de method GiveBlockSpecs maken we een picturebox aan, deze picturebox geven we values en returnen we naar de form.

```
4 references
class Blocks
{
    // create Random object
    public Random rng = new Random();

    // create location variables
    static int xLocation = 200;
    static int yLocation = 0;

    1 reference
    public PictureBox GiveBlockSpecs()
    {
        xLocation = 200;
        yLocation = 0;
        PictureBox Block = new PictureBox();
        Block.Location = new Point(xLocation, yLocation);
        Block.Size = new Size(50,50);
        Block.BackColor = RandomColor();
        return Block;
    }

    2 references
    public void UpdateLocation(int xLocationModifier, int yLocationModifier, PictureBox Block)
    {
        xLocation = xLocation + xLocationModifier;
        yLocation = yLocation + yLocationModifier;
        Block.Location = new Point(xLocation, yLocation);
    }
```

In de form stellen we de waarden die we uit de Blocks class hebben gekregen gelijk aan de block uit de form class. Daarna voegen we de block toe aan de form zodat je hem ook visueel kunt zien.

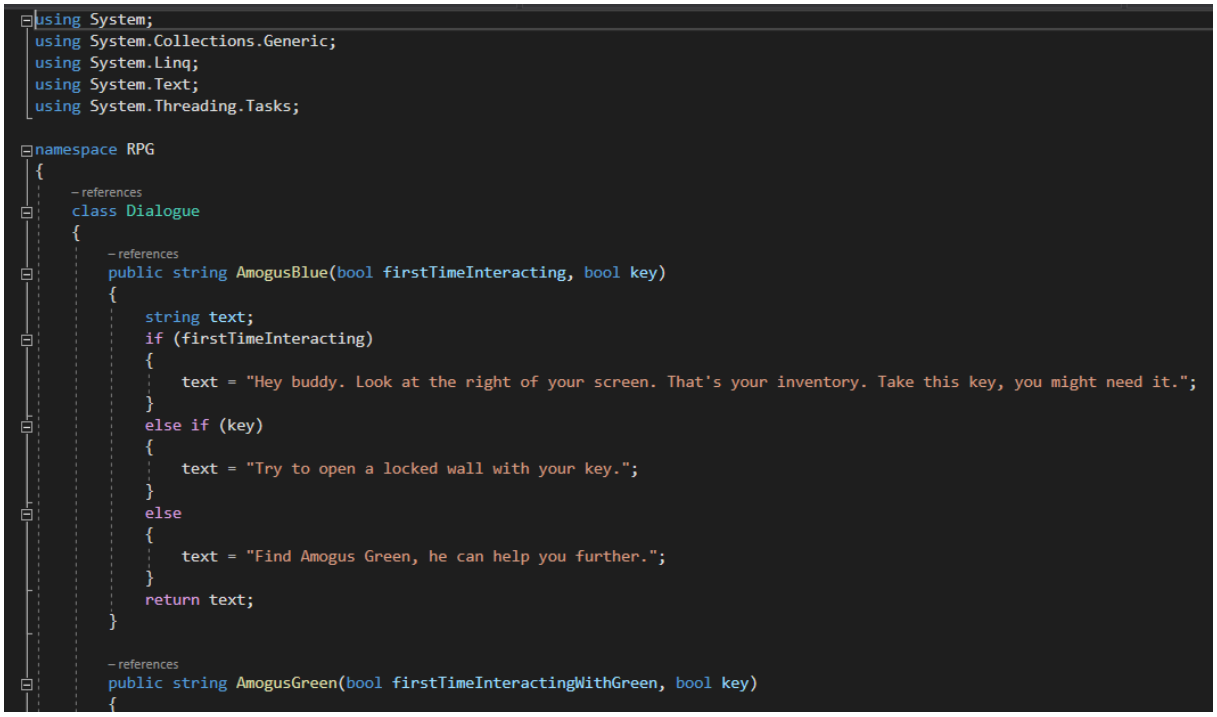
```
// gets the picturebox from the Blocks.cs class and adds it to the form
2 references
public void CreateBlockInForm()
{
    // get the block
    block = blocks.GiveBlockSpecs();

    // add the block
    Controls.Add(block);
}
```

Voor als je je afvraagt waarom ik in de Blocks class steeds een nieuwe picturebox aanmaak, ik zou ook die regel weg kunnen halen en de block van de form meegeven. Maar als ik dat doe dan is er altijd maar één block actief. Op deze manier kun je de oude block die je niet meer bestuurt nog wel zien in de form.

3.1.7 Scheiding GUI en Domain

Ik neem als voorbeeld de RPG game.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RPG
{
    - references
    class Dialogue
    {
        - references
        public string AmogusBlue(bool firstTimeInteracting, bool key)
        {
            string text;
            if (firstTimeInteracting)
            {
                text = "Hey buddy. Look at the right of your screen. That's your inventory. Take this key, you might need it.";
            }
            else if (key)
            {
                text = "Try to open a locked wall with your key.";
            }
            else
            {
                text = "Find Amogus Green, he can help you further.";
            }
            return text;
        }

        - references
        public string AmogusGreen(bool firstTimeInteractingWithGreen, bool key)
        {
```

Bovenaan kun je zien dat ik geen using.winforms gebruik dit geld voor elke class in dit programma.

3.1.8 leesbaarheid / onderhoudbaarheid (feedback van anderen is belangrijk)

Ik heb met Ruben samen gewerkt aan GeoBlastr, we vonden allebei van elkaar dat we duidelijke comments bij de code zetten. Zelf als ik terug kijk in mijn eigen code dan kan ik ook begrijpen wat ik allemaal heb gedaan.

3.2 Eigen project

<Hier beschrijf je het door jou gemaakte eigen project (wedstrijd) met daarin de volgende onderdelen:

- *Een beschrijving van je project, eventueel met relevante code snippets.*
- *Indien van toepassing: Een flowchart.*
- *De gebruikte bronnen*

Het RPG project zou je wel kunnen zien als het wedstijd project, daarin laat ik alle leeruitkomsten naar voren komen behalve constructors.

Het is een soort van avontuur game. Je begint in een bibliotheek waarin je moet praten tegen een blauw karakter om verder te kunnen. Hij vertelt je hoe je verder moet. Vervolgens kom je op een grasveld. Hier heb je 2 karakters en een mes. Jij mag kiezen wat je doet. Er zijn 6 verschillende endings die je kunt krijgen. Wat er allemaal op het veld gebeurt werd na een tijdje zo verwarrend dat ik maar een flowchart heb gemaakt. Deze flowchart zit als bestand in repos in "Bijlages". Daar kun je hem in het groot bekijken.

Code is erg simpel. De dingen die ik vooral heb gebruikt zijn Procescmd ding voor de key inputs, en een timer om elke milliseconde te checken of er dingen moeoten gebeuren. Verder heb ik alles netjes in classes gezet. Method overloading kun je terugvinden in de Dialogue class. Je krijgt verschillende dialogen gebasseerd op of je een bepaalde item hebt of niet. Encapsulation kun je terug vinden in de classes met fields. Kijk maar is naar de Inventroy class. Alle fields zijn daar private. Verder is de gui en de domain goed gescheiden. Je ziet geen forms zooi of andere zooi in de classes die daar niet horen. En voor return/parameter eigen return waarde mag je kijken bij FRM_Field > TMR_GameTime_Tick > // AMOGUS PINK of bij // AMOGUS YELLOW > // interaction with knife. Hier zie je dat als de speler met een mes in de inventory een karakter aanspreekt hij een ander dialoog krijgt.

Alles is zo een beetje aangetoond in FRM_Field. FRM_Library was meer een kleine intro dat je geforceerd wordt je praten met karakters zodat je weet dat dat later moet. En ik wilde de inventory gebruiken. Dat heb ik in FRM_Library ook gedaan met de key.

FRM_Meeting is hardcoded zooi. Dat zit er meer in zodat het er wat mooier uit komt te zien. Deze hele game was meer een grap en de meeting maakte dat compleet.

Ik probeerde ook nog een bossfight erin te zetten. Daar zou je dan belanden als je levend uit field zou ontsnappen. Maar mijn laptop vond 4 forms in een project niet zo leuk.

Bronnen: Among Us, Minecraft, W3schools.
Deze topic: https://stackoverflow.com/questions/37342384/collision-detection-between-two-picture-boxes-not-working-c-sharp-form

4 Reflectie / evaluatie

4.1 Waar ben ik trots op?

De vooruitgang die ik heb gemaakt. Ben trots op de programma BattleSim en DinoGame. Echt 2 prachtige programma's leuk concept ook niet lelijk gedesigned. Sirted is een programma waar ik een soort van Tetris probeer te maken. De reden dat ik dit wilde maken is, ik wilde altijd al TetrisAttack namaken maar dat lukte nooit. En normale tetris maken is al een goed begin. RPG, topprogramma laat het iedereen spelen die ik zie. En astroblastr is helemaal perfect. Superleuk speel het altijd als we op locatie zijn.

4.2 Wat doe ik een volgende keer anders?

Ik maakte eerst geen crc-kaarten. Blijkt uiteindelijk best handig te zijn als je even van tevoren nadenkt wat je wilt gaan maken. Maar als ik nu terugkijk zou ik niet iets anders doen. Misschien nog wat eerder feedback vragen.

4.3 Welke formatieve indicatie zou ik mezelf geven voor de oriëntatie Software?

Leeruitkomst oriëntatie Software		
Onderdeel	Criterium	Rating
Aandacht voor algoritmiek	Je kunt eenvoudige applicaties schrijven die stapsgewijs oplossingen voor problemen vinden door het uitvoeren van logische testen en eenvoudige stapsgewijze berekeningen.	*G/O, eenvoudige applicaties gaan me makkelijk af. Kijk bijvoorbeeld naar de orientatie training opgaves, alles wat ik had gemaakt lukte gewoon. En bij de training opgaves bij de verdieping lukte ook wat ik wilde maken plus ik gaf er nog mijn eigen draai aan.*
Basisvaardigheden	Je begrijpt en past de volgende programmeerconcepten toe:Variables, conditional statements, loops, methods, lists/ arrays en enum's.	*G, ben nog wel wat matiger met enums, heb het meerder malen geprobeerd te gebruiken maar dan switch je uiteindelijk naar een array ofzo. Alle andere concepten begrijp ik wel en kan ik toepassen.*
Aantonen	Je vraagt feedback van een docent en laat zien dat je deze feedback verwerkt hebt.	*G, ik heb regelmatig gesprekken met mijn semestercoach over hoe ik ervoor sta. Ik laat zien wat ik heb gemaakt en of het voldoende de leeruitkomst aantoont.*

4.4 Welk verdiepend(e) profiel(en) kies ik en waarom?

Software, ik vind logica heel leuk. Heb plezier in het coderen.