Design rapport - Informatik

Lavet af:
Carl Dyresberg og Emil Andersen.

Klasse
1.4

Lærer
Simon Ingemann Christiansen

Indholdsfortegnelse

Introduktion

Igennem forløbet har vi arbejdet med at fremstille et nytænkende produkt som skulle programmers og testes. Igennem processen har vi arbejdet med at lave en problemstilling og stille spørgsmål til hvad vi gerne ville opnå med produktet. En af de ting der fangede os var griefing scenen i videospil. Nogen videospil er halvt ødelagt pga. folk som udnytter mindre vidende greifere og får dem til at download deres trojanske heste. Vi valgte at fokusere på et open-source hack som ville blive hostede på GitHub eller andre sidder hvor folk nemt kan tilføje deres egne koder eller bare downloade scriptet og bruge det.

Kravspecifikationer

Problemstilling
I denne designproces vil vi arbejde med hacks og scripts til computerspil. Det skal gøre et givent spil nemmere for brugeren og gøre det mere sikkert for brugeren grundet vores open-source idé og det hele skal så appellere til målgruppen “griefers og greifing”.

Målgruppe
Vores målgruppe er “griefers og greifing”. Det er folk som foretrækker høj spil status frem for spillefærdigheder og spil udforskning. De kan identificeres som “hackere” i daglig tale, og bøjer generelt spillets regler og regelsæt til deres egen fordel, for at gøre spillet nemmere for dem selv. I skemaet nedenfor kan målgruppen “griefers og greifing”, betegnes som en blanding af “Killers” og “Achievers”.

Målgruppen i forhold til vores produkt er hovedsageligt børn fra 10 - 17 år som spiller computerspil, der kræver præcision og hurtige reaktions-evner. Børnene har et ønske om at opnå en bedre spille status hurtigere gennem "snyd".

Produktdesign

Produkt Idéer
Vores proces startede med at vi opstillede en problemstilling og valgte hvem vi ville appellere til og det satte "basen" for vores brainstorm. Vi gennemgik spil som vi kunne lave assisterende værktøjer til for at hjælpe/bekæmpe den døende greifing scene. Blandt de spil var:
Roblox, League of legends og Zombs Royale.
Vores endelige valg til prototypen blev Zombs Royale.

Efter vores brainstorm skulle vi lave overvejelser om hvordan det skulle assistere brugeren. Det skulle være et værktøj som gør brugeren tæt på uovervindelige eller bare giver dem en større fordel til at opnå højere spil status, og derfor valgte vi at fokusere på at give dem en større fordel grundet de mere aggressive værktøjer har en tendens til at "ødelægge" spillet for de normale brugere, hvor værktøjet kan hjælpe de mindre erfarne brugere uden at ødelægge det for de andre spillere.

Begrundelse for valg af målgruppe og produkt
Vi valgte specifikt at arbejde på et assisterende værktøj til Zombs Royale fordi vi i forvejen havde gjort os nogle opdagelser om spillet. Ud fra det fandt vi ud af at folk har lagt mange "hacks" op til det. Vi scannede nogen af filerne og fandt ud af de fleste var trojanske virusser og nogle få som bare ikke virkede. I sidste ende ville det gå ud over brugeren fordi at de måske ikke er klar over der er en trojansk virus i filen og det kan ende med at koste dem filer, penge osv.

Produkt Prototype

Første udkast
Vi startede med et første udkast for at få en forståelse af hvordan vi ville komme med vores endelige produkt. Det var et meget simpelt system som kun kiggede efter pixel farver på karaktererne i spillet. Det fungerede godt, men vi ville gerne undersøge nogle flere muligheder såsom image recognising.

```
1  from PIL import ImageGrab
2  import time
3  import pyautogui
4
5  lookcolor = 255,255,255
6  time.clock()
7  image = ImageGrab.grab()
8  for y in range(0, 100, 10):
9      for x in range(0, 100, 10):
10         color = image.getpixel((x, y))
11         if color == lookcolor:
12             print(time.clock(),"\n", color, "\n", x,",",y, "\n")
13
```

De første 3 linjer importerer nogle moduler vi bruger. Videre til linje 3 hvor vi definerer hvad for en farve vi søger efter og i det her tilfælde hvid. Linje 6 sætter en timer til som vi brugte til at optimere scriptet og så tager linje 7 et screenshot af skærmen som den så tjekker senere. Linje 8 til 10 går gennem hver tiende pixel i en 100x100 box og for hver omgang loopet køre definerer den "color" til farvekoden af den pixel den kigger på og så til sidst tjekker den i linje 12 til 13 om farven er hvid. Vi ville så til sidst erstatte print linjen med en som rykker musen hen på pixelen som den fandt, og fordi den køre i et loop fungere det ligesom et aimlock.

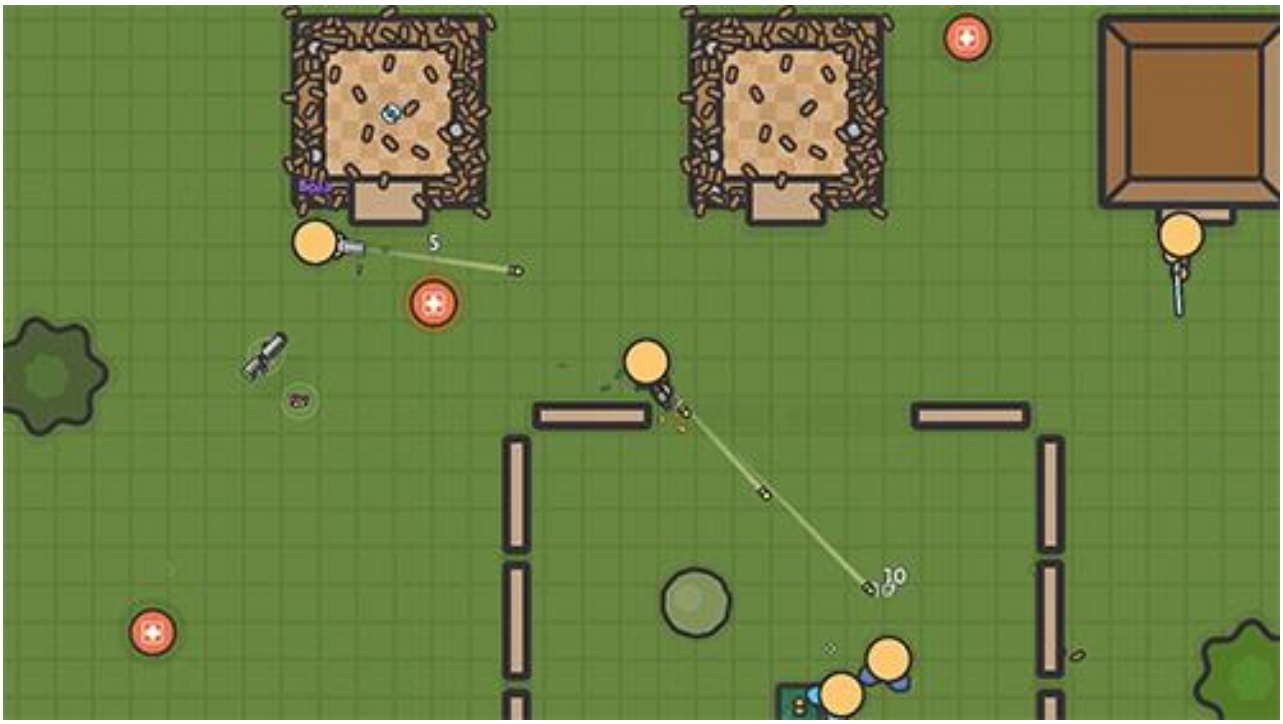
Andet udkast
Vi arbejde også på en måde hvor vi kunne udføre det gennem image recognising. Det endte med at denne metode ville være i gennemsnit 6-8 millisekunder langsommere end vores pixel-finder hvilket ikke ville være særlig effektivt, men vi kom frem til at vi kunne bruge det her til at lave en aimlocker i stedet for en aimbot. Aimlocken holder musen på en spiller i modsætning til en aimbot som finder spilleren på skærmen. Vi fremstillede et lille program med pyscreeze modulet, som fandt pixelsearchet og det fungerer rimelig godt.

```
1  import pyscreeze
2  import time
3
4  def search(image):
5      time.clock()
6      x,y,useless1,useless2 = pyscreeze.locateOnScreen(image)
7      print(x,y,'| clocked: ', time.clock())
8
9
10 try:
11     image = 'playerpart.png'
12     search(image)
13 except:
14     print("Error, Failed to find image")
15
```

Linje 1 til 2 importere pyscreeze som vi bruger til image recognising og time som vi bruger til at tage tid. Selve programmet starter ved linje 3 og der sætter vi en funktion op som tager billeder og søger efter det. Den første linje i funktionen starter timeren, den næste leder efter billedet ved brug af pyscreeze og den sidste printer koordinaterne og printer hvor lang tid det tager. Fra linje 10 til 12 prøver den først at definere billedet og bruger "search" funktionen. Hvis der er en "exception" så printer den, at billedet ikke kunne findes pga., når billede ikke kan findes så outputer pyscreeze en error i stedet for bare at output "none".

Valget mellem vores to udkast

Vi havde valget mellem vores to udkast som begge havde fordele og ulemper. Vores andet udkast brugte image recognising som ledte efter en del af karakteren hvilket gør, at hvis fjenden har et kostume på så ville vi have svært ved at finde fjenden, men det kunne potentielt være mere pålideligt fordi den tjekker efter mere end bare én pixel. Det første udkast ville måske låse sig fast på et træ eller en bygning men den ville være hurtigere og den ville have en større chance for at låse sig fast på en fjende med et kostume på. Ud fra det ville vi nok fokusere på udkast nummer 1 grundet den er hurtigere og kan måske låse sig fast på folk med kostumer hvilket er en fordel siden de fleste spillere har kostumer.



Billede af Zombs Royale kampsituation, uden kostumer

https://imgur.com/a/nUCMIgM

Speedet op video af en af vores udkast som så leder efter bildækket på baggrunden og giver os koordinaterne af bildækket og hvor lang tid det tog.

Produkt Evaluering

Måden vi evaluerede vores prototype på, var gennem en spørgeundersøgelse. Vi kunne ikke få folk til at teste vores prototype direkte, men vi tilpassede vores spørgsmål til målgruppen griefers. Hvis de udspurgte viste interesse for vores produkt og idé, ville vi kunne færdiggøre vores prototype, og måske hvis der var interesse for det kunne vi realisere det gennem et "open source-projekt" på GitHub og muligvis arbejder videre på værktøjet til Zombs Royale og udvide til andre spil.

Spørgsmål	Svarmuligheder	Besvarelser (ud af 35 besvarelser)
Har du brugt assisterende værktøjer (hacks) i spil?	Ja Nej	9 26
Hvis du har brugt assisterende værktøjer i spil, har du så nogensinde fået en virus eller lign af det?	Ja Nej Ved ikke	5 2 2
Kan du lide spillere der bruger assisterende værktøjer til spil?	Ja Nej Ligeglad	7 23 5
Synes du det fortjent at folk som leder efter de værktøjer måske ender med at få en virus af det?	Ja Nej Ligeglad	18 6 11
Ville du kunne finde på at bruge et assisterende værktøj til et spil hvis chancen for at ende med en virus var mindre?	Ja Nej Måske	8 25 2
Er status og gode resultater vigtige for dig i spil?	Ja Nej	27 8

Undersøgelsesresultater

Spørgeundersøgelse resultater
Vores spørgeundersøgelse gav meget forskellige resultater. Folk var mest imod brug af assisterende værktøjer til hjælp i spil. Der var dog nogen der viste interesse for vores idé, som også selv bruger assisterende værktøjer i spil. Det er dog svært at sige om et assisterende værktøj som vores prototype, ville appellere til lige netop dem som besvarede vores spørgeskema grundet vores prototype er baseret på Zombs Royale. Zombs Royale er ikke et særligt udbredt spil, de fleste spillere er børn som ikke har lige så meget kendskab til assisterende værktøjer og andre former for hacks.

Fejlkilder
Spillet Zombs Royale er ikke spillet af så mange. Derfor er interessen for vores produkt ikke særlig stor. Hvis vi skulle have flere til at synes om vores produkt, skulle vi have valgt et mere populært spil. Men da populære spil er bedre sikret, ville processen være nærmest umulig for vores evner som programmører.
Der er skins, eller kostumer i spillet som kan gøre detektion af fjendtlige, spiller svært for "boten". Dette kan resultere i at programmet ikke vil fungere optimalt, sagt på en anden måde, det ville være nytteløst. Vi kom dog ikke frem til en løsning på dette problem.
Man kan selvfølgelig også bare have glemt at installere den rigtige version af Python.

Konklusion

Igennem forløbet har vi arbejdet med at fremstille et nytænkende produkt som skulle programmers og testes. Igennem processen har vi arbejdet med at lave en problemstilling og stille spørgsmål til hvad vi gerne ville opnå med produktet. En af de ting der fangede os, var griefing scenen i videospil. Vi valgte at fokusere på et open-source hack som ville blive hostede på GitHub eller andre sidder hvor folk nemt kan tilføje deres egne koder eller bare downloade scriptet og bruge det.

Ud fra denne rapport kan vi konkludere, at hvis vi havde muligheden og tiden til at arbejde videre på vores produkt ville vi gerne prøve at lære et nyt programmeringssprog som er hurtigere og bedre optimeret end Python. Python er generelt et langsomt programmeringssprog, men kan være brugbart i mange scenarier bare ikke vores scenarie. Udover det ville vi gerne udvide vores rækkevidde og lave assisterende værktøjer til flere spil og til sidst lave det hele open-source så folk kan hjælpe med at tilpasse og optimere værktøjerne til deres behov og krav. Endnu en grund til at lave det hele open-source er at så kan vi få mere tillid fra folk fordi de kan kigge koden igennem og undgå at få virus som de fleste fra vores undersøgelse har prøvet at få.