DESIGN DOCUMENTATIE

IVY BAREND 3012556 INTERACTION DESIGN KERNMODULE 1

TRIP WATCH

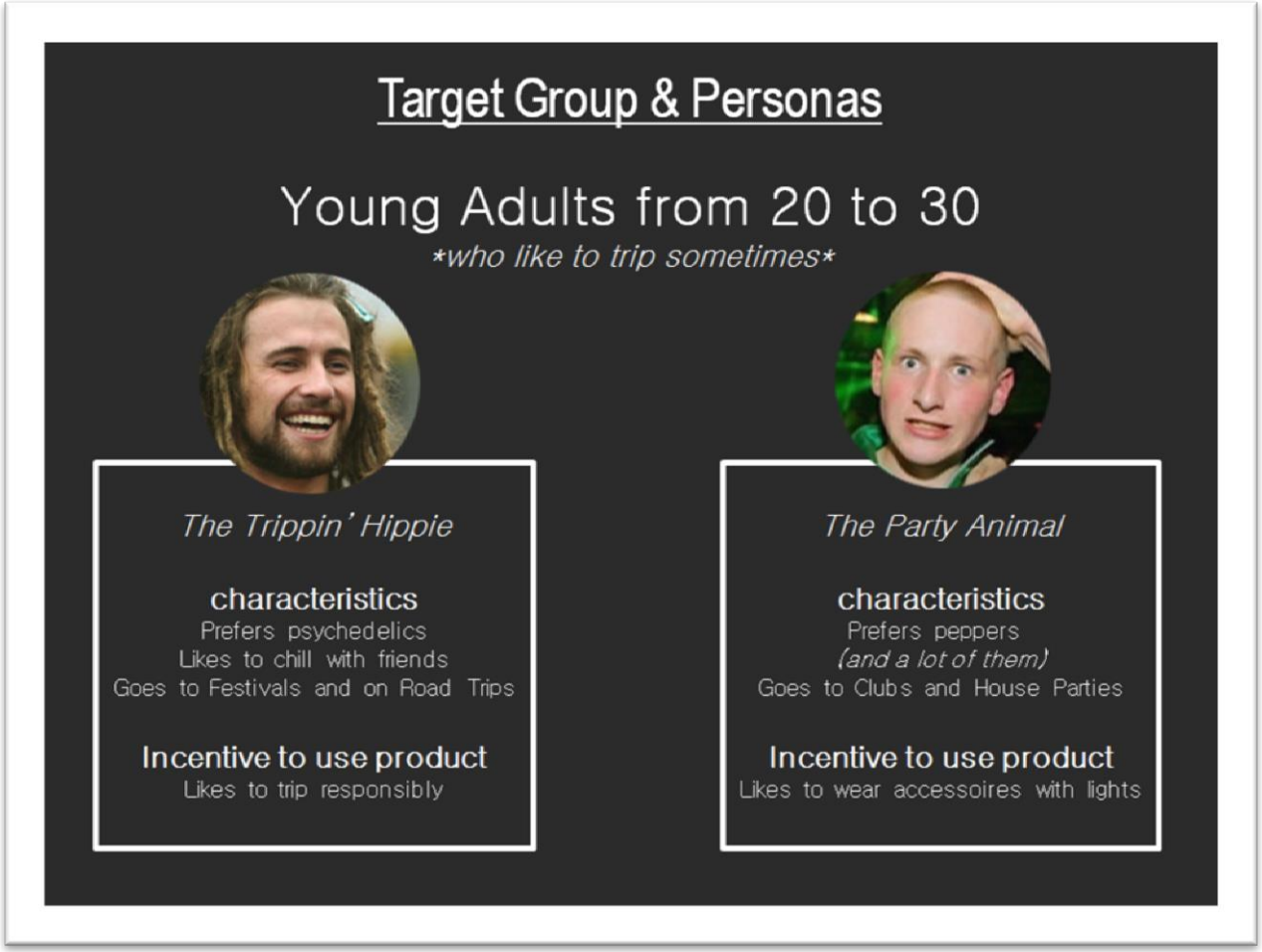
CONCEPT

TripWatch is een armbandje voor de verantwoordelijke feestganger. Het helpt je tóch op je lichaam te kunnen letten wanneer je natuurlijke zintuigen daar even niet meer in staat toe zijn. Door middel van bewegingsregistratie houdt de TripWatch voor je bij hoeveel je al gedanst hebt, en hoeveel vocht je daarbij hebt opgebruikt. Dit communiceert de TripWatch naar de gebruiker met simpele kleur-communicatie. Verder hoef je als gebruiker niets te doen, want wanneer je wat drinkt registreert hij dit ook en zal hij zich weer resetten.

Aanleiding concept

Het thema wat we de allereerste les hebben meegekregen rondom deze kernmodule was 'drinken'. We kregen de opdracht om een aantal snelle concepten uit te werken rondom dit thema. Ik dacht na over momenten in mijn eigen leven wanneer het belangrijk is om te drinken, en van alles wat ik had bedacht viel mijn oog het meest op het idee van de TripWatch.

Even in het kader van eerlijkheid: bijna iedere student maakt wel eens gebruik van recreationele drugs. Daar val ik onder, en dus is het idee voor dit project gekomen vanuit een intrensieke behoefte om verantwoordelijk te trippen. Een van de gevaarlijkste dingen, vooral in de zomer, rondom drugs is dat je vaak niet meer op de hoogte bent van wat je lichaam wil. Soms is dat fijn, betekenend dat je de hele dag en nacht lekker door kan dansen, maar het maakt het ook een gevaarlijke activiteit. Veel van mijn generatie in ieder geval weten dit wel, maar omdat besef van tijd ook maar een relatief begrip is in de staat waarin je dan bent, is het heel makkelijk om te vergeten op tijd even wat water te drinken.



Daarom leek de TripWatch het interessantst voor mij om uit te werken. Niet alleen omdat ik er graag eentje zou willen hebben zelf, maar ook omdat het vanuit een Design perspectief leerzaam en interessant is. Product -en interactie ontwerp voor de nuchtere mens zijn al veel richtlijnen voor, maar ontwerpen voor iemand met een altered state of mind is een ander verhaal.

Om me hier in verder te helpen heb ik al vrij vroeg een aantal persona's rondom mijn doelgroep gemaakt, om een duidelijk beeld te krijgen van de situatie waarin mijn doelgroep van gebruikers zich kunnen vinden.

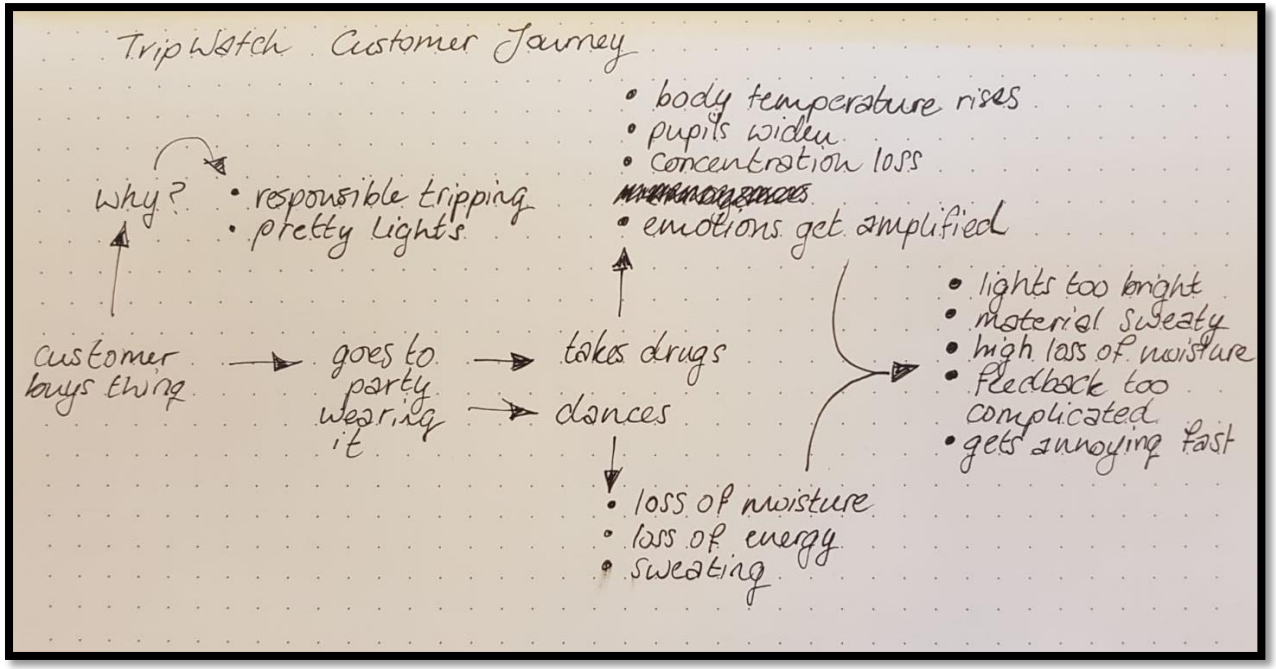
Hierop gebaseerd heb ik voor mezelf een aantal doelen gesteld rondom het interactie ontwerp van de TripWatch.

- Zo min mogelijk ingewikkelde interactie met het ding
- Visuele aantrekkelijkheid is een must
- Geen tekst of tekens als feedback
- Het moet comfortabel zijn een geen obstakel vormen tijdens de activiteiten

Functionaliteit

Op basis hiervan besloot ik een aantal ontwerp stappen te maken. Ik begon met een kleine Customer Journey-achtige brainstorm omtrent de ervaring. Hierdoor werden de obstakels in het ontwerp al iets duidelijker.

De functionaliteit van de TripWatch is in principe heel simpel. De kunst bij dit ontwerp is om het juist zo non-ingewikkeld mogelijk te houden. Wat het doet, is een geschatte waarde van de hoeveelheid vocht in het lichaam van de gebruiker communiceren. Het moet niet aanvoelen als een alarm, maar als een herrinering. Gebruik maken van geluid heeft dus geen zin, het zou opgevat kunnen worden als irritant, of helemaal niet worden opgemerkt vanwege het omgevingsgeluid en de eventuele audiotieve hallucinaties van de gebruiker op dat moment.



De grootste uitdaging rondom dit concept zit ‘m in de vormgeving, en daarmee de interactie die de vormgeving uitnodigt of afstoot. Mensen aan de drugs zijn op zijn zachtst gezegd nogal gevoelige typjes, dus de vormgeving is hierbij erg belangrijk.

Wat verder nog wel in me op kwam was de verhoogde lichaamstemperatuur. Het zou interessant zijn als de TripWatch dit op de een of andere manier zou kunnen meten om dat vervolgens mee te nemen in de calculaties omtrent het vochtgehalte in het lichaam. Echter zag ik geen oplossing waarbij het resultaat accuraat genoeg zou zijn, en hanteerbaar binnen het idee van een horloge-achtig apparaat, dus liet ik dit idee even links liggen, om me te focussen op de hoofdinteractie.

Vormgeving

Zoals in het Customer Journey brainstorm dingetje te zien is, ben ik rondom de vormgeving een aantal obstakels tegen gekomen. Ik wilde gebruik maken van kleurcommunicatie, omdat letters en tekens niks doen of onleesbaar zijn door de invloed die veel recreationele middelen op je ogen kunnen hebben, leek kleurcode me de meest simpele, veelomvattende en begrijpelijke manier om de informatie naar de gebruiker te communiceren. Daarbij zijn dingen met lichtjes vaak heel populair op festivals en feestjes. Echter, is het wel belangrijk dat de lichtintensiteit niet te hoog is. Een full-brightness LED is voor de nuchtere mens soms al moeilijk om naar te kijken, laat staan iemand wiens pupillen drie maal zo groot zijn als normaal.

Verder is het belangrijk dat het materiaal waarvan het armbandje gemaakt is, comfortabel zit. Ik weet uit ervaring dat veel soorten armbandjes en horloges ontzettend gaan schuren en plakken wanneer je helemaal bezweet op een feestje staat, dus het materiaal moet zo min mogelijk draagirritaties veroorzaken. Emoties worden voor sommigen nogal vergroot in zo’n situatie, en dus kunnen kleine irritaties opeens voelen alsof het ’t meest verschrikkelijke ding is wat je ooit hebt meegemaakt. Ik wil dus dat de armband van een zacht materiaal gemaakt is wat op zijn minst niet extra zweet veroorzaakt of gaat schuren.

Voor de kleurcommunicatie ben ik gegaan voor een tint die van groen naar rood afloopt. Dit omdat dit een universeel herkenbaar teken is van goed naar slecht, en dus makkelijk herkenbaar voor iedereen in wat voor staat dan ook. Volgens de kleurenwetten is de kleur die tussen groen en rood zit geel, maar ik heb er bewust voor gekozen om het van groen naar paars naar rood te laten verlopen. Dit omdat binnen mijn eigen ervaring althans, de kleur geel niet ‘indrukwekkend’ genoeg is binnen een feestsetting. Het valt weg in alle andere kleuren en lampen die bijvoorbeeld op een festival of feestzaal aanwezig zijn, en lijkt daarom bijna wit.

Verder maak ik maar gebruik van 3 kleuren om de feedback zo simpel mogelijk te houden. Mijn product is misschien aantrekkelijk voor de doelgroep omdat er coole lampjes op zitten, maar als ik het visueel te druk maak wordt het originele doel van mijn product allicht uit het oog verloren door de gebruiker.

PROTOTYPE

Voor de opdracht van Kernmodule 1 waren de eisen om 1 sensor en 1 actuator aan elkaar te hangen en daarmee een simpel systeem te bouwen. Het uitgangspunt hiervan was arduino en openFrameworks, maar omdat mijn concept in zijn ideale vorm een draadloos ding is, leek het me handiger om het voor realisme in ieder geval gewoon in de arduino IDE te doen.

Verder is me geadviseerd om de interactie zo klein mogelijk te houden, maar omdat ik wel de kern van de interactie er in wilde hebben, heb ik toch besloten om meer dan één functionaliteit toe te voegen.

Het prototype kan detecteren wanneer iemand aan het dansen is, en wanneer het vochtlevel op is kan het prototype ook detecteren of er gedronken wordt. De main loophole hierin is natuurlijk dat de gebruiker ook kan drinken voordat het vochtlevel helemaal op is, en daarmee eigenlijk de loop moet resetten, maar om dat zo accuraat en vloeiend mogelijk te kunnen neerzetten was een vrij grote opgave die eigenlijk niet veel meer te maken had met het doel van deze kernmodule.

Dus om het doel van het uitlezen van een sensor en daarmee een actuator aan te sturen, te bereiken, en tegelijkertijd de extrensieke interactie van het ding zo vloeiend mogelijk te kunnen maken in de gegeven tijd, heb ik besloten om de interactie even lineair te houden. Meer hierover wordt duidelijk in het software-gedeelte van het document.

Hardware

De hardware van het TripWatch-prototype bestaat uit een GY-521 Gyroscoop/accelerometer chip, en een RGB LED. Door het uitlezen van de GY-521 kan de TripWatch detecteren wanneer iemand hele heftige bewegingen maakt in een korte tijd, en daarmee aan de hand van een aantal parameters concluderen dat de persoon aan het dansen is. Met deze sensor kan de TripWatch ook de beweging van dansen detecteren.

Wanneer de persoon aan het dansen is, gaat het vochtlevel van het lichaam omlaag. Dit vertaalt de TripWatch naar de RGB led door de kleur van groen naar rood te laten verlopen met een snelheid gerelateerd aan de hoeveelheid vocht die je verliest tijdens het dansen. De RGB LED gaat in het prototype van groen naar rood via de kortste weg: dus wel via geel. Dit is puur om het qua code-structuur leesbaarder te maken voor mezelf, en de kern, van groen naar rood, is alsnog wel aanwezig.

Zie het filmpje voor een demonstratie van het prototype.

Software

De Code maakt gebruik van de Wire library, omdat de GY-521 gebruikt maakt van de I2C Bus om met de arduino te kunnen communiceren.

Allereerst worden de waardes van de Accelerometer en de Gyroscoop uitgelezen, samen met een benodigde correctie. Waarom die correctie nodig is ben ik niet achter gekomen, maar het werd aangeboden als oplossing voor een probleem wat ik ondervond met de lezingen, en het scheen te werken.

Na iedere loop met lezingen wordt er 1 waarde bij de var readCount opgeteld, om bij te houden hoeveel loops hij al heeft doorlopen.

```
void loop() {
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU,14,true);

  int AcXoff,AcYoff,AcZoff,GyXoff,GyYoff,GyZoff;

  //Acceleration data correction
  AcXoff = -950;
  AcYoff = -300;
  AcZoff = 0;

  //Gyro correction
  GyXoff = 480;
  GyYoff = 170;
  GyZoff = 210;

  //read Acc data
  AcX=(Wire.read()<<8|Wire.read()) + AcXoff;
  AcY=(Wire.read()<<8|Wire.read()) + AcYoff;
  AcZ=(Wire.read()<<8|Wire.read()) + AcYoff;


  //read gyro data
  GyX=(Wire.read()<<8|Wire.read()) + GyXoff;
  GyY=(Wire.read()<<8|Wire.read()) + GyYoff;
  GyZ=(Wire.read()<<8|Wire.read()) + GyZoff;

  readCount++;
```

Dan wordt er per lezing gecheckt of er een extreme verandering heeft plaatsgevonden. Dit was niet heel overzichtelijk te doen in arduino code, maar wat deze functie doet is basically checken of er bij zowel de gyroscoop of de accelerometer een verandering heeft plaatsgevonden op alle Axis, van minimaal de waarde van dancingBM, wat 3000 is in deze code.

Wanneer die voorwaarden worden bereikt, wordt er 1 opgeteld bij highValCount.

```
//if gyro vals differ startval + dancing benchmark, high val is read
if (((GyX > GstillX + dancingBM || GyX < GstillX - dancingBM) &&
     (GyY > GstillY + dancingBM || GyY < GstillY - dancingBM) &&
     (GyZ > GstillZ + dancingBM || GyZ < GstillZ - dancingBM)) ||
    ((AcX > lastAcX + dancingBM || AcX < lastAcX - dancingBM) &&
     (AcY > lastAcY + dancingBM || AcY < lastAcY - dancingBM) &&
     (AcZ > lastAcZ + dancingBM || AcZ < lastAcZ - dancingBM)))
{
  highValCount++;
}
```

```
//when 30 readings are done;
if (readCount > 30)
{ //if more than half the readings are high, dancing is true
  if (highValCount > 0.5 * readCount)
  {
    dancing = true;
  } else {
    dancing = false;
  }//reset the reading vals
  highValCount = 0;
  readCount = 0;
}
```

Na dat er 30 lezingen zijn gedaan, worden readCount en highValCount met elkaar vergeleken. Wanneer meer dan de helft van de lezingen extreem zijn, neemt de TripWatch aan dat je aan het dansen bent, door de bool ‘dancing’ op true te zetten. Wanneer het tegenovergestelde waar is, staat de bool ‘dancing’ op false.

Er zit dus een delay van maximaal 10 seconden tussen het stoppen/beginnen met dansen en de detectie ervan. Ik heb hier voor gekozen om te voorkomen dat de TripWatch een range van andere bewegingen gaat zien als daadwerkelijk continuëus dansen.

```
if (dancing)
{
  if (waterLevel > 0){
  waterLevel--;
  if (waterLevel > 50){
    r = map(waterLevel, 100, 50, 0, 255);
  } else {
    g = map(waterLevel, 0, 50, 0, 255);
  }
} else {
  dehydrated = true;
  }
}
```

Als ‘dancing’ true is, gaat de var waterLevel naar beneden. In de realiteit zal deze waarde veel ingewikkelder zijn dan het nu is. waterLevel zou moeten worden beïnvloed door een berekening van de algemene snelheid waarmee een gemiddeld persoon water verliest tijdens een activiteit als dansen, en ook door de buitentemperatuur waar de gebruiker zich bevind. Dit was echter niet deel van de kern van de interactie dus heb ik er voor gekozen om het in het prototype simpel te houden. Verder zou het demonstreren van de interactie realistisch gezien veel te lang duren.

Verder kun je zien dat ik de waarde van waterLevel door middel van de map() functie direct vertaal naar de output van de RGB LED. WaterLevel begint op 100 (%). Wanneer deze boven 50 is, gaat de rode LED langzaam aan, en daarna gaat de groene led langzaam uit.

Wanneer waterLevel op 0 komt, gaat de bool dehydrated op true, betekenend dat de gebruiker moet gaan drinken.

```
if (dehydrated){
  waterLevel = 0;
  blinkRed();
  if (pitch > lastPitch){
    difference += pitch - lastPitch;
    if (difference > 60){
      waterLevel = 100;
      r = 0;
      g = 255;
      dehydrated = false;
    }
    lastPitch = pitch;
  } else {
    difference = 0;
  }
  lastPitch = pitch;
}
```

Zolang dehydrated true is, blijft waterLevel op 0 en knippert de RGB LED rood. Vervolgens gaat de TripWatch checken voor de beweging van drinken. Dit doet hij d.m.v. de var ‘pitch’, die de waardes van de accelerometer pakt om uit te rekenen wat de hoek van de chip is t.o.v. de grond. Na een aantal testrondes kwam ik tot de conclusie dat de beweging van drinken uit een glas of fles neerkomt op een minimaal verschil van ongeveer 60 graden. Elke loop vergelijkt hij de recente waarde met de laatste waardevan pitch, en wanneer dat groter is dan 60, heeft de gebruiker gedronken en kan alles worden gereset. Zo niet wordt de var ‘difference’ gereset.