Feedback na gesprek met Dries De Roeck

Anner Tiete <anner@beyond.io>

di 6/01/2015 11:40

Postvak IN

Aan: EVERAERT Jan (s) <jan.everaert@student.ehb.be>;

CC: Kevin Verelst <kevin@beyond.io>; Dries De Roeck <dries.deroeck@gmail.com>; VAN DEN BROECK, Wouter <wouter.van.den.broeck@ehb.be>;

Hi Jan,

Nog eens onze beste wensen voor 2015 - het gaat zonder twijfel een heel boeiend jaar worden!

Omdat ik er voor de kerst-break zelf niet actief bij kon zijn, even dit mailtje om alles wat er besproken is met ons en Dries nog eens op een rijtje te zetten.

Even nog wat pointers :

- We denken dat het een goed idee is om eerst de "state of the art" grondig te onderzoeken, zodat je weet wat goede voorbeelden zijn, wat er al reeds gedaan is door anderen en welke problemen deze concepten of producten al opgelost hebben. (Waarschijnlijk heb je dit in tussentijd al gedaan.) Zorg dat je dit ook allemaal mooi bijhoudt en catalogiseert, zo past deze research meteen in je verslag of dossier later. Ideaal zou zijn als je de verschillende andere gelijkaardige projecten met elkaar vergelijkt op een meer schematische manier (plus en min punten opsommen of op een grafiek of in een tabel) Het is vooral de bedoeling dat je uit deze research "steelt" wat nuttig is voor jouw werk!
- Zorg eerst voor een goed uitgeschreven / geschetst verhaal van het spel dat je wil gaan ontwikkelen voor je in de technische details duikt (hoe leuk dit hacken ook kan zijn) - misschien kan je bepaalde technische problemen makkelijk uit de weg gaan als je je verhaal lichtjes anders formuleert. Dit helpt ook erg bij het vooraf definiëren van wat je wel en wat je niet gaat doen. Zodra dat bepaald is, is het werk dat je te wachten staat al veel duidelijker en valt dit veel beter te plannen. De uitwerking hiervan hoef je niet als iets 'af' te bekijken, het is zelfs aan te raden het op een niet digitale manier te doen om zo sneller feedback van je doelgroep te krijgen (bvb: via een low-fidelity prototype of wizard of oz methode)
- Definieer deelproblemen (bvb: hoe zorg ik voor "loops" in mijn programma), bedenk voor deze deelproblemen onafhankelijk van elkaar een aantal oplossingen en combineer deze dan stuk per stuk tot complete concepten - hier is een klassieke "morfologische kaart" een erg handig hulpmiddel voor (simpel voorbeeldje met een fiets : https://btgroepi.wordpress.com/morfologische-kaart/ - jij kan dit iets abstracter doen, maar probeer altijd snel een schetsje te maken, dit helpt erg goed bij het visualiseren en communiceren in latere fases)
- Bekijk altijd je systeem (wat er gebeurd) en je technische (hardware) oplossing als 1 ; bouw ze gelijk op en test ze ook gelijktijdig in een vroeg (on-af) stadium - het zou zonde zijn moest je hardwarematig (of softwarematig) al heel erg gevorderd zijn en dan ontdekken dat je bepaalde dingen helemaal niet nodig hebt of ze niet te combineren vallen. Weer een argument om de bovenstaande techniek van de morfologische kaart met deelproblemen en oplossingen te gebruiken. Door op deze manier te werken vermijd je tevens je te laten beperken door bepaalde technologische elementen.

Succes en zoals je weet kan je ons altijd benaderen voor feedback of advies via Slack!

Anner Tiete

anner@beyond.io
+32 485 61 31 05

beyond.io

www.beyond.io
Gijzelaarsstraat 10, 2000 Antwerpen
Belgium