Healed

Équipe :

CHICOINE Grégory - Game Designer
FRIES VALENTIN - Développeur
LEMOINE Adrien - Game Design - chef de projet
MILANO Vincent - Développeur
PELTE Maxime - Développeur

en partenariat avec:

agorize benext

Introduction :

Healed est un jeu vidéo d'aventure plateformer 2D sur téléphone et pour le web visant à apporter du réconfort aux personnes suivant un traitement médical en leur apportant des bonus/une monnaie à chaque fois qu'ils suivent une séance thérapeutique.
Le jeu est accessible pour tout le monde, mais les personnes ayant un suivi médical peuvent avoir un profil unique à leur maladie et ainsi obtenir certains avantages particuliers.
Le jeu se veut sans difficulté, le principe repose sur le fait que le challenge se déroule dans la vraie vie (traitement) et que la récompense se trouve dans le jeu.

Le concept :

L'idée de base du jeu est d'attribuer au patient jouant à *Healed* un certain nombre «d'étoiles» à la fin de chaque traitement. Par exemple, une personne atteint de la mucoviscidose reçoit chaque jour après sa séance de kinésithérapie respiratoire cette récompense.
Pour l'obtenir, l'hôpital en collaboration propose un QR Code à flasher via l'application en récompense à la suite du traitement. Cela intègre les étoiles au jeu et le joueur peut alors les

Mécanique du jeu et Core Gameplay :

Boucles de gameplay :

Le joueur contrôle un avatar évoluant dans une zone. Il peut dépenser les différentes ressources qu'il a récoltées dans le jeu ou suite à un traitement pour débloquer des portes et accéder au reste de la carte.
Objectif → Challenge → Reward (boucle du core gameplay)

Objectif = découvrir le reste de l'univers

Challenge : (cette partie est spécifique, le jeu ne souhaite proposer au joueur qu'une suite de rewards)
● Le challenge est le traitement que suit le patient
● Le challenge in-game est de parvenir jusqu'à la porte et d'avoir suffisamment d'étoiles pour payer le coût de l'ouverture

Reward : accès à une nouvelle partie du monde

Boucles de mécanique :

Action → Simulation → Feedback (boucle de la mécanique du passage d'une porte)

Action : Le joueur arrive près d'une porte (exemple : un rocher à détruire), un message lui détaillant l'action qu'il s'apprête à effectuer et le coût de celle-ci apparaît alors.

Simulation :
● Le joueur accepte de payer : son coût est soustrait de son total d'étoiles cumulés et la porte disparaît, ouvrant le passage en au joueur.
● Le joueur refuse de payer ou n'a pas les moyens : rien ne se passe et le message disparaît.

Feedback : La porte disparaît, le compteur d'étoile est mis à jour, le joueur peut à présent avancer dans cette zone.

Les différents types de profil :

Si le joueur possède une maladie et un traitement particulier, il spécifie lors de la création de son compte la maladie qu'il a. Dans ce cas là, un système de QR Code distribué aux hôpitaux confirme la maladie du patient, à qui il suffit de photographier le code pour cela.
Selon le type de maladie, le joueur possède une mécanique qui lui est unique : si le patient souffre de la mucoviscidose, alors son avatar peut effectuer un double saut, contrairement aux autres profils. Ils peuvent ainsi atteindre des zones qui leur sont réservées grâce à cette mécanique.
Un joueur sans maladie peut se créer un compte, mais ne possède pas de mécanique spéciale. De ce fait pour gagner des étoiles qui lui permet d'évoluer, il doit lire et s'intéresser à différents panneaux donnant des informations sur les maladies en général. Ces panneaux sont dispatchés dans l'univers du jeu et sont accessibles par tous.
Il y a aussi un quizz posant des questions sur les maladies en question qui donnera des étoiles en fonction du résultat obtenu.

3C :

Controllers = touches fléchées pour se diriger à gauche et à droite, la barre d'espace pour sauter
En arrivant vers une porte, appuyer sur une touche pour confirmer (touche E) et une pour décliner (touche P). Possibilité de pouvoir tchatter avec les autres joueurs à l'intérieur de l'auberge, lieu online dédié à la discussion.

Character : le joueur contrôle un personnage. celui-ci ne peut pas mourir. De plus avec l'argent collecté il peut acheter des tenues différentes pour ce personnage.

Camera : caméra en perspective suivant le personnage centré, pour un side view scroller.

3M :

Objectif à court terme : explorer l'univers.

Objectif à moyen terme : découvrir de nouvelles zones, accumuler les différentes types de monnaies.

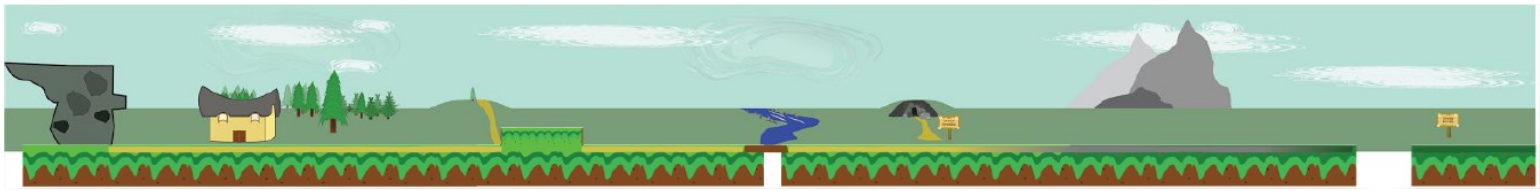
Objectif à long terme : découvrir l'ensemble de l'univers, avoir accumulé tous les trophées de succès.

Level Design :

Cette partie là représente le début du jeu. Le joueur démarre près de l'auberge qui est un lieu de rencontre entre les différents joueurs. Le but est d'en faire un lieu d'échange pour les joueurs afin de discuter de leurs expériences, etc.

À partir de ce plateau le joueur peut rejoindre d'autre lieu de l'univers, en se mettant devant un panneau, et en validant la destination.

A long terme nous souhaiterions proposer un large pannel de lieux différents que le joueur pourrait découvrir au fur et à mesure de son avancée.



Marketing et Sponsors :

Préproduction :

Campagne de financement par crowdfunding, en association avec les différentes association collaboratives (ex : les virades de l'espoir)

Publicité in-game :

Possibilité d'inclure des bandeaux de publicité

Les boutiques :

À l'intérieur du jeu le joueur peut de plusieurs façon acheter certains objets. Il peut acheter une monnaie in-game lui permettant d'acheter de nouvelles tenues,
Le joueur peut aussi récolter à travers le jeu des bons cumulables leur permettant d'avoir des produits particuliers en partenariat avec les sponsors.

Développement du projet :

Organisation : Git & Slack

BitBucket a été la plateforme d'hébergement Git choisie pour ce projet : elle permet de créer des équipes disposant de dépôts privés, adaptés à cette semaine de Hackathon. Deux dépôts différents ont été créés :
● **healed-documentation** (https://bitbucket.org/healed/healed-documentation) : contient toute la documentation du projet ansi que certains assets de la charte graphique.
● **healed-project** (https://bitbucket.org/healed/healed-project) : contient les sources du projet Healed.
Concernant la méthodologie de développement, un Git flow classique a été adopté afin d'isoler chacune des fonctionnalités durant les phases de conception, évitant les conflits lors de leur développement.
De manière à faciliter la communication, centraliser les échanges et permettre un suivi aisé du développement, une team Slack a été mise en place : https://healed-esgi.slack.com/ . Connecté au dépôt BitBucket du projet, le suivi des commits de l'équipe permettait d'avoir une vue d'ensemble en temps réel du travail des développeurs, facilitant l'avancement général du projet. Il s'agissait de plus d'un moyen simple de s'échanger les sprites et textures utilisées en jeu.

Afin d'éviter la redondance du code côté API, deux classes génériques ont été développées : **GenericController.js** et **GenericRouter.js** . Ainsi, la majorités des actions de CRUD ont pu être simplifiées de la route jusqu'à l'interaction avec la base de données

Architecture & Technologies :

Nous avons orienté nos choix technologiques vers une fullstack JavaScript afin d'assurer une meilleure compatibilité avec tous les supports Web :
● **Node.js** : serveur principal (express.js) servant le contenu statique et une API REST servant d'interface avec une base de données MySQL.
● **AngularJS** : framework front-end MVVM délivrant une interface dynamique au navigateur et interagissant avec l'API Node.js.
● **Babylon.js** : framework WebGL permettant la mise en place d'un environnement 2D/3D dans un canvas HTML5.
Côté Angular (front), une organisation par *modules* (grandes fonctionnalités) a été favorisé plutôt qu'une organisation par *rôles* (contrôleurs, modèles, …). Les fonctionnalités sont ainsi correctement isolées dans l'architecture même du projet, permettant aux développeurs de trouver plus facilement ce qu'ils cherchent quelle que soit l'ampleur prise par le projet.
Afin d'éviter la redondance du code côté API, deux classes génériques ont été développées : **GenericController.js** et **GenericRouter.js** . Ainsi, la majorités des actions de CRUD ont pu être simplifiées de la route jusqu'à l'interaction avec la base de données