UE 223

Groupe 5

Rendu final
BANDIT MANCHOT

Masson Tiffany

Mazeau Thomas

Pierre Gabrielle

Vacher Théo

Table des matières

Présentation de la méthode de travail :

Partage des documents :

Avant de commencer le travail, nous avons décidé de choisir un moyen de se partager les documents. Notre choix s'est évidemment porté sur GitHub. Ce fut d'abord compliqué de saisir toutes les fonctionnalités, mais une fois qu'on a tous bien assimilé l'utilisation de Git, le partage des documents était bien plus intuitif que si on avait simplement envoyé les fichiers à chaque modification.

Réunions :

On a utilisé Discord pour se tenir au courant des avancées de chacun, ainsi que pour organiser nos réunions vocales. Nous avons fait environ une réunion tous les deux jours afin de faire le point sur les parties non fonctionnelles, et établir la liste des tâches restantes.

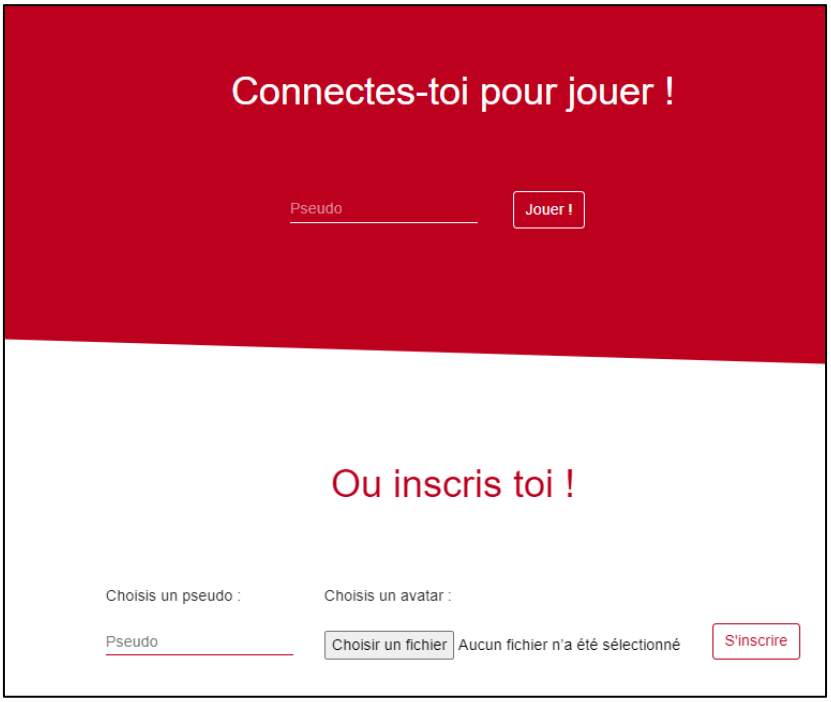
Répartitions des tâches

Tâches	**Elève**
JavaScript et CSS du Bandit Manchot, mise en commun des documents	Tiffany Masson
CSS des pages, affichage selon connexion dans index.php, documentation	Thomas Mazeau
Création de la base de données et communication avec les pages php, mise en commun des documents et réalisation du rapport	Gabrielle Pierre
Communication avec la base de données, et débogage de certaines fonctionnalités.	Théo Vacher

Présentation générale du projet

Notre mini-projet doit porter sur le jeu Bandit-manchot. Nous avons donc commencé par chercher les règles du jeu. On a donc discuté du fonctionnement de celui-ci selon les consignes données. Nous en avons conclu qu'il fallait créer une base de données qui répertorie les pseudos et leurs scores. Ensuite, nous avons fait le choix de faire une page d'accueil qui permet à l'utilisateur de se connecter, et de jouer au jeu. Nous avons également ajouté une page pour les règles du jeu, et une page pour les crédits.

Capture d'écran de la page d'accueil :



Nous avons intégré 4 niveaux de difficulté du jeu :

- 3 rouleaux et probabilité de jackpot haute,
- 3 rouleaux et probabilité de jackpot faible,
- 4 rouleaux et probabilité de jackpot haute,
- 4 rouleaux et probabilité de jackpot faible.

Il a fallu ensuite réfléchir au système de score. Le score s'affichera sous forme de somme d'argent (en clochettes). Un nouveau joueur va commencer le jeu avec 100 clochettes automatiquement. La partie coûte 10 clochettes au niveau 1, 20 clochettes au niveau 2, 30 clochettes au niveau 3 et 40 clochettes au niveau 4. L'utilisateur gagnera une somme selon l'item sur lequel il tombe. Les sommes sont calculées en fonction de la probabilité de tomber sur l'item. Evidemment, plus l'item apparait dans le rouleau, plus le joueur peut tomber 3 fois sur cet item, et donc gagnera une plus petite somme de clochettes. Nous avons choisi le personnage Kéké (ou le sac de clochettes pour la version avec des fruits) comme étant l'item « Jackpot ».

Exemple du niveau 1 :

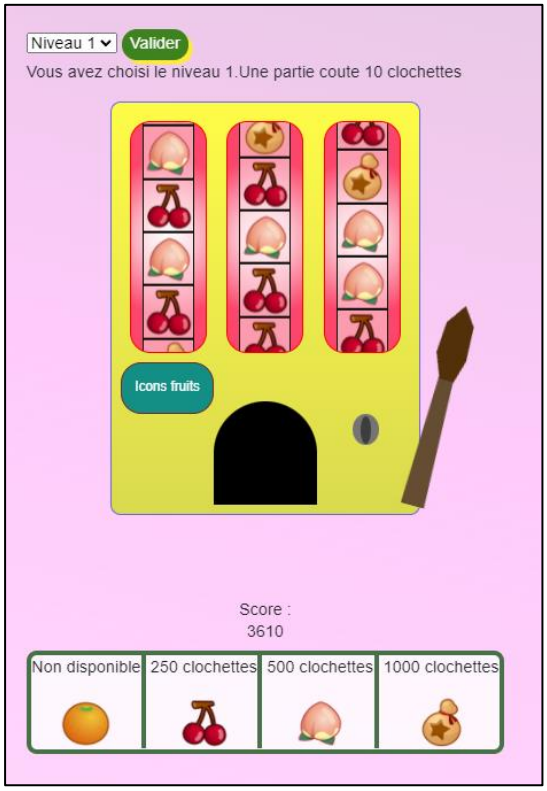
Quant aux items, nous avons pensé à mettre des icônes en rapport avec Animal Crossing, puisque nous avions défini la monnaie comme étant la clochette. L'utilisateur peut d'ailleurs choisir un design avec des personnages d'Animal Crossing, ou avec les fruits du même jeu.

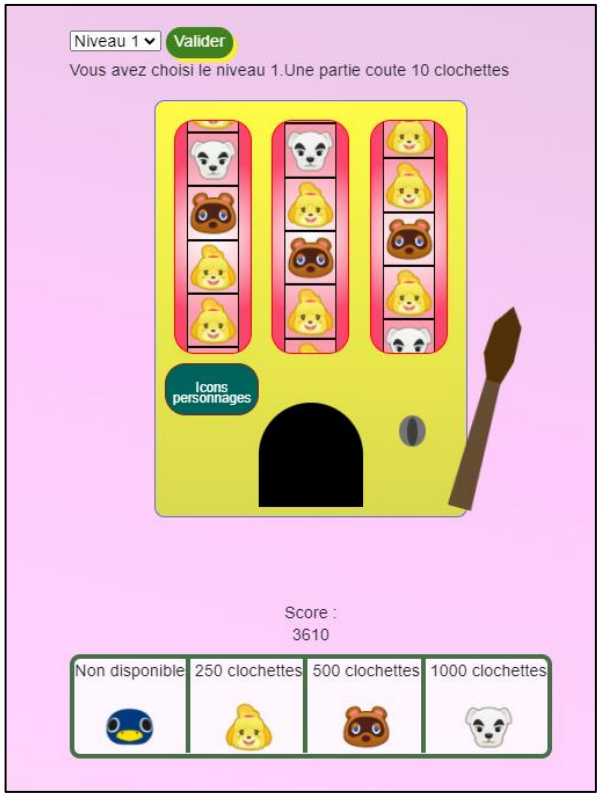
Quand on parle de machine à sous, on pense au son de pièces qui tombent et autres bruitages liés à ce jeu. On a donc décidé de rajouter des sons : lorsque l'utilisateur met une pièce dans la machine on a un son de pièce qui tombe, et lorsque l'utilisateur perd ou gagne il y a un bruitage qui rappelle la défaite ou la victoire.

On peut penser également pour les couleurs à une palette de type « casino », avec du vert foncé, du rouge, du doré… On s'était orientés vers cette palette de couleur au début. Mais dans l'évolution de notre projet, on a préféré s'aventurer dans quelque chose de plus original. Etant donné qu'on avait choisi l'univers d'Animal Crossing pour les items et la monnaie, on a voulu faire quelque chose dans le même thème.

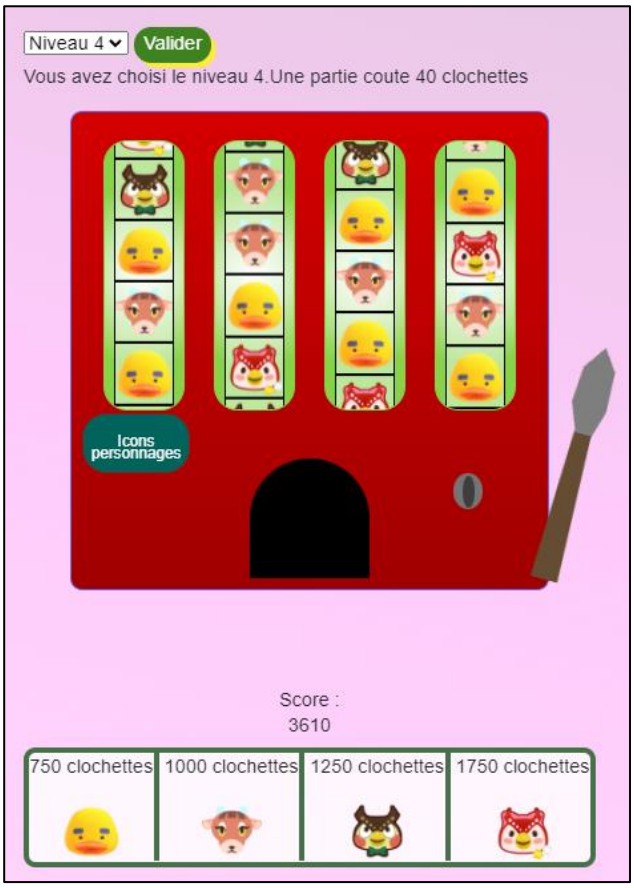
L'utilisateur peut également choisir une photo de profil qui sera incorporée dans la base de données.

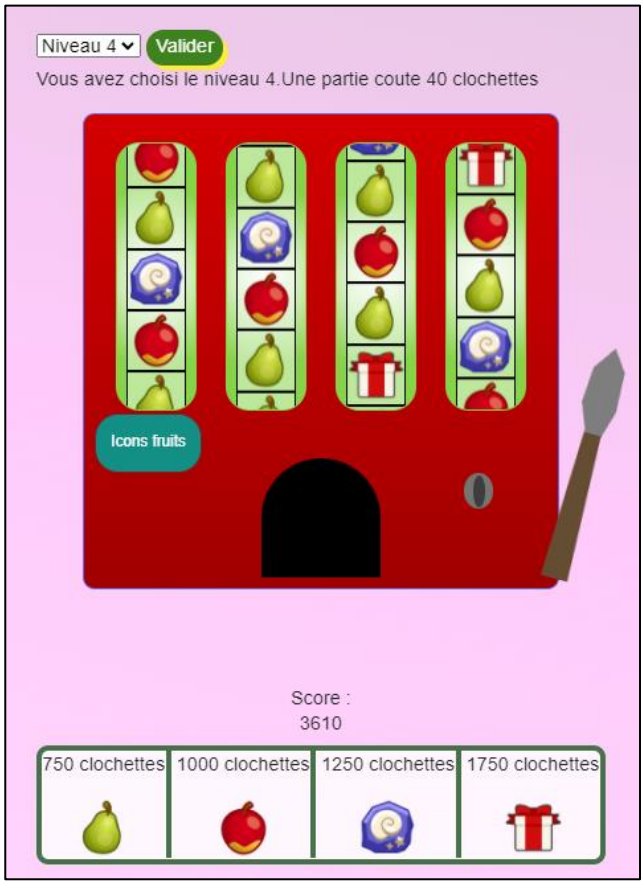
Visuel du niveau 1 selon les icônes choisies :



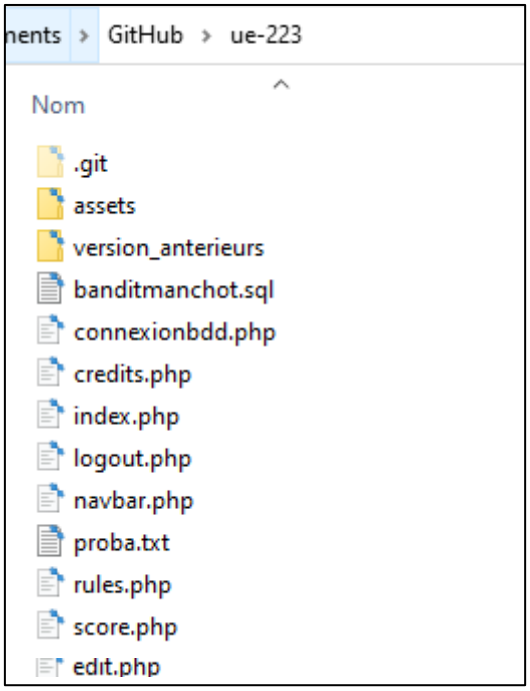


Visuel du niveau 4 selon les icônes choisies :





Explication approfondie du code



« banditmanchot.sql » le code de la base de données.

« Proba.txt », explication des chances de gagner.

« connexionbdd.php » permet aux fichiers php de communiquer avec la base de données (BDD).

« score.php » permet de modifier le score de l'utilisateur dans la BDD, pour payer la partie, être remboursé en cas d'erreur et récupérer la somme gagnée en cas de victoire.

« rules.php » page des règles du jeu.

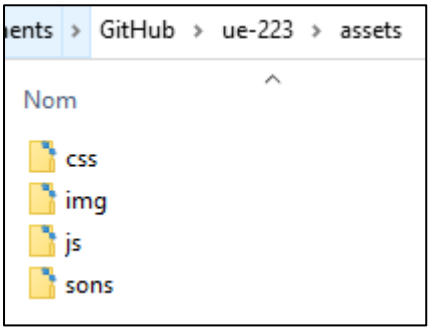
« credits.php » nos noms.

« logout.php » permet de se déconnecter.

« edit.php » permet de modifier ou supprimer son compte.

« navbar.php » contient le code du haut de page avec le logo, l'accès aux trois pages, afficher l'utilisateur connecté avec sa photo de profil et son nom, ainsi que le bouton déconnexion.

« index.php » contient deux affichages, si l'utilisateur n'est pas connecté on affiche un formulaire de connexion. Si l'utilisateur est connecté on affiche l'espace d'affichage du bandit manchot. L'affichage du bandit manchot dépend du niveau choisis.



« css » contient style.css, « img » contient les images du code, l'icône du site, les icônes du bandit manchot, les icônes de curseur et le gif de pièces pour la victoire.

« js » contient script.js et « sons » contient les sons mp3 du bandit manchot.

Page connexionbdd.php :

C: > Users > Tiffa.LAPTOP-7UK61KJ2 > Documents > GitHub > ue-223 > connexionbdd.php

```php
<?php
  try
    {
      $bdd=new PDO('mysql:host=localhost:3306;charset=utf8;dbname=banditmanchot','root','');
      array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION);
    }
    catch(Exception $e)
    {
      die('Erreur : ' . $e->getMessage());
    }
?>

```

Le code permet la connexion à la base de données « banditmanchot » avec l'identifiant « root » et un mot de passe vide. La variable « $bdd » sera appelée dans les autres fichiers pour communiquer avec la BDD.

Page index.php :

```php
<?php
  $page="home";
  ini_set('display_errors', 1);
  include 'connexionbdd.php';
  session_start();
?>
```

On récupère le fichier connexionbdd.php, on définit que la page actuelle est « home » et `display_errors` permet d'afficher les erreurs.

Sessions_start() permet de récupérer/créer une session.

```html
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- style -->
    <link rel="icon" type="image/png" href="assets/img/logo.png"/>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
    integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
    crossorigin="anonymous">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.
    min.css" integrity="sha512-Fo3rlrZj/k7ujTnHg4CGR2D7kSs0v4LLanw2qksYuRlEzO+tcaEPQogQ0KaoGN26/
    zrn20ImR1DfuLWnOo7aBA==" crossorigin="anonymous" referrerpolicy="no-referrer" />
    <link rel="stylesheet" href="assets/css/style.css">

    <!-- titre -->
    <title>Bandit Manchot - Accueil</title>
  </head>

  <body>

  <?php
```

Puis on déclare la page html (dans <meta> on appelle la balise pour utiliser @media pour le responsive et la compatibilité avec Edge), les styles (donc notre fichier style.css et l'utilisation de notre icone de page) et l'utilisation de Bootstrap et une police d'écriture.

```php
<?php
display_errors Aa ab .* ? sur 1

if (isset($_POST['jouer'])) {
    if (isset($_POST['login'])) {
        $_SESSION['login'] = strip_tags((String) $_POST['login']);

        if ($_SESSION['login'] != "") {

            $exists = false;

            $requeteScore = $bdd->prepare('SELECT pseudo, score, img FROM utilisateur WHERE pseudo = ?
            ');
            $requeteScore->execute(array($_SESSION['login']));

            while ($data = $requeteScore->fetch()) {
                if ($_SESSION['login'] == $data['pseudo']) {
                    $exists=true;
                    $_SESSION['score'] = $data['score'];
                    if ($data["img"] != "")
                    {
                        $_SESSION['urlImage'] = '' . base64_encode($data["img"]
                        ) ;
                    }
                    else {$_SESSION['urlImageD'] = 'assets/img/icon_Clochettes.png';} //image par
                    defaut
                }
            }

            $requeteScore->closeCursor();

            if (!$exists) {
                $requeteEcriturePseudo=$bdd->prepare('INSERT INTO utilisateur (pseudo, score) VALUES
                (:pseudo, :score)');
                $requeteEcriturePseudo->execute(array(
                'pseudo' => $_SESSION['login'],
                'score' => 100,
                ));
                $requeteEcriturePseudo->closeCursor();
                $requetePseudoCree = $bdd->prepare('SELECT pseudo, score FROM utilisateur WHERE pseudo
                = ?');
                $requetePseudoCree->execute(array($_SESSION['login']));

                while ($data = $requetePseudoCree->fetch()) {
                    if ($_SESSION['login'] == $data['pseudo']) {
                        $exists=true;
                        $_SESSION['score'] = $data['score'];
                    }
                }

                $requetePseudoCree->closeCursor();
            }
        }
    }
}

include('navbar.php');
if (!isset($_SESSION['login'])) {
?>
    <section id="login-section" class="play d-flex justify-content-center align-items-center">
      <h1 class="login-title">Connectes-toi pour jouer !</h1>
      <form id="form" action="index.php" method="POST">
        <div id="msgPseudo"></div>
        <input id="pseudo" type="text" placeholder="Pseudo" name="login" required>
        <input class="btn btn-login-submit btn-primary-outline" type="submit" value="Jouer !"
        name="jouer"></input>
      </form>
    </section>
<?php
} else {
?>
<main class="main-container-jeu d-flex flex-column align-items-center justify-content-center">
    <h1 class="text-center pb-5">Bandit Manchot !</h1>
    <div class="w-50 game flex-fill d-flex justify-content-center">
    <span class="align-self-center">
      <div id="formChoixNiveau" > <!-- method="POST"> -->
        <select id="choixNiveau" type="select" name="chxNiveau">
```

Ensuite, en php on vérifie si le formulaire de connexion a été remplis. A la troisième ligne de la capture à gauche, on a sécurisé les données du pseudo en vérifiant qu'il s'agisse bien d'une chaîne de caractères (pas de HTML).

S'il est rempli, la variable de session 'login' récupère le nom du joueur, s'il n'est pas nul, on vérifie si ce login existe dans la BDD.

S'il existe, on récupère le score du joueur.

Sinon on le rentre dans la BDD, et on lui donne 100 clochettes de départ.

Puis on passe à l'affichage de la page, on récupère le fichier « navbar.php » pour afficher le haut de page.

Si l'utilisateur n'est pas connecté, on affiche le formulaire de connexion.

Sinon on affiche le bandit manchot.

```
<div id="formChoixNiveau" > <!-- method="POST"> -->
  <select id="choixNiveau" type="select" name="chxNiveau">
      <label>Quel niveau de difficulté ?</label>
      <option value="1" selected="selected">Niveau 1</option>
      <option value="2">Niveau 2</option>
      <option value="3">Niveau 3</option>
      <option value="4">Niveau 4</option>
  </select>
  <button id="choixNiveauButton" ><!-- type="submit"> -->Valider</button>
</div>
<p id="reponseChoix"></p>

<div id="banditLvl1Container" class="banditContainerNone">
    <div id="banditLvl1Corps">
        <div id="banditLvl1Rouleau1">
            <div class="rouleau1 rouleau1Anim">
                <span class="r1els r1el02"></span>
                <span class="r1els r1el04"></span>
                <span class="r1els r1el03"></span>
                <span class="r1els r1el02"></span>
                <span class="r1els r1el02"></span>
                <span class="r1els r1el03"></span>
                <span class="r1els r1el02"></span>
                <span class="r1els r1el02"></span>
                <span class="r1els r1el03"></span>
                <span class="r1els r1el02"></span>
                <span class="r1els r1el03"></span>
                <!-- retour -->
                <span class="r1els r1el02"></span>
                <span class="r1els r1el04"></span>
            </div>
        </div>
        <div id="banditLvl1Rouleau2">···
        </div>
        <div id="banditLvl1Rouleau3">
            <div class="rouleau3 rouleau3Anim">···
            </div>
        </div>
    </div>
    <div id="banditLvl1AnimePieces"></div>
    <div id="banditLvl1Manche"></div>
    <span id="choixIcons">Icons personnages</span>
    <span class="fenteAPiece1"><span></span></span>
</div> <!-- end of .banditLvl1Container -->

<div id="banditLvl2Container" class="banditContainerNone">···
</div> <!-- end of .banditLvl2Container -->
```

```
        <div id="banditLvl4Container" class="banditContainerNone">···
        </div> <!-- end of .banditLvl4Container -->
        <div>
            <h1 id="resultat"></h1>
        </div>
        <p id="messageErreur"></p>
        <div id="explications">
            <div><p>100 clochettes</p><span  class="explic r1el01"></span></div>
            <div><p>250 clochettes</p><span  class="explic r1el02"></span></div>
            <div><p>500 clochettes</p><span  class="explic r1el03"></span></div>
            <div><p>1000 clochettes</p><span  class="explic r1el04"></span></div>
        </div>
    </span>
  </div>
  <span class="score">Score : <h1><?php echo $_SESSION['score']; ?></h1></span>
  <!-- scripts -->
    <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
  <!-- JS -->
  <script src="assets/js/script.js" async></script>
</main>
<?php
  }
?>
<ul class="circles">
  <li><img src="https://cdn-icons-png.flaticon.com/512/247/247837.png" width="60" height="60"/></li>
  <li><img src="https://cdn-icons-png.flaticon.com/512/218/218362.png" width="25" height="25"/></li>
  <li><img src="https://cdn-icons-png.flaticon.com/512/247/247814.png" width="50" height="50"/></li>
  <li><img src="https://cdn-icons-png.flaticon.com/512/218/218362.png" width="100" height="100"/></li>
  <li><img src="https://cdn-icons-png.flaticon.com/512/247/247814.png" width="100" height="100"/></li>
  <li><img src="https://cdn-icons-png.flaticon.com/512/247/247837.png" width="75" height="75"/></li>
  <li><img src="https://cdn-icons-png.flaticon.com/512/247/247814.png" width="150" height="150"/></li>
  <li><img src="https://cdn-icons-png.flaticon.com/512/218/218362.png" width="200" height="200"/></li>
  <li><img src="https://cdn-icons-png.flaticon.com/512/247/247837.png" width="25" height="25"/></li>
  <li><img src="https://cdn-icons-png.flaticon.com/512/218/218362.png" width="50" height="50"/></li>
</ul>
  <!-- scripts -->
  <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
<!-- scripts
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X
+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script> -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy"
crossorigin="anonymous"></script>
</body>
</html>
```

Le bandit manchot se présente avec d'abord le choix du niveau, puis le corps des niveaux.

Le corps des niveaux présente 3 ou 4 rouleaux, chaque rouleau contient 11 cases « r1els » puis les classes « r1el01 » à « r1el18 » sont l'icone de la case.

Les deux r1els supplémentaires sont la pour l'animation avant partie et disparaissent quand le jeu commence.

Après les rouleaux, on retrouve la div « anime pieces » qui affiche le gif de pièces en cas de victoire, ainsi que la div « banditLv1Manche » qui correspond au manche pour commencer la partie, le bouton pour changer les icones du bandit manchot, fruits ou personnages, puis une fente à pièce pour le design.

Après les corps des niveaux, on retrouve l'affichage du résultat, qui est vide par défaut.

Ensuite l'espace pour les messages d'erreurs, qui peut contenir un message indiquant que vous ne pouvez pas payer la partie.

En dessous on retrouve l'explication de la signification des icones du jeu, qui est mis à jour pour chaque niveau et chaque changement d'icônes.

En sortant du corps du bandit manchot on retrouve le score du joueur, qui ne se met pas à jour pour le moment.

Ensuite l'appel du jquery et de notre fichier script.js.

Enfin on récupère le bas de page commun, avec l'animation d'icones en font de page et l'appel des scripts nécessaire pour leur animation.

Pages credit.php et rules.php :

```php
<?php
  $page="credits";
  ini_set('display_errors', 1);
  include('connexionbdd.php');
?>
```

```php
<?php
  $page="rules";
  ini_set('display_errors', 1);
  include('connexionbdd.php');
?>
```

Elles commencent avec le php pour définir la $page, et avec le html de début de page, avec le titre correspondant, et l'appel du navbar.php. La page credit.php contient nos noms, tandis que la page Rules.php contient l'explication du jeu puis les probabilités de victoires.

Pages logout.php :

rs > Tiffa.LAPTOP-7UK61KJ2 > Documents > GitHub > ue-223 > logout.php

```php
<?php

  // Starts session
    session_start();

  // Destroys session
  session_destroy();

  // Redirects to index.php
  header('Location: index.php');
```

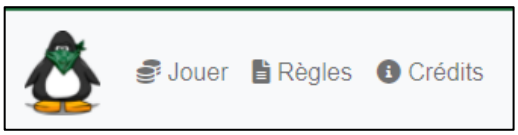
Permet la déconnection de l'utilisateur et reste sur la page index.php.

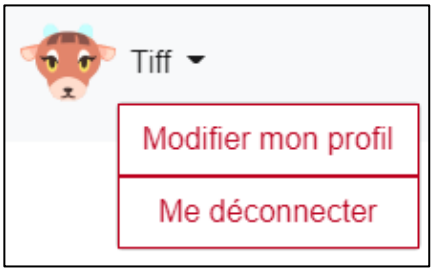
Page navbar.php :

```php
<nav class="fixed-top navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="index.php">
    <img class="img-fluid" src="assets/img/manchot.png" alt="Logo" height="50" width="50">
  </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
  data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false"
  aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">

      <li class="nav-item">
        <a class=         <?php if($page=='home'){ ?>
              "nav-link active"
          <?php } else { ?>
              "nav-link"
          <?php } ?>
          href="index.php"><i class="fas fa-coins"></i> Jouer
        </a>
      </li>

      <li class="nav-item">
        <a class= <?php if($page=='rules'){ ?>
              "nav-link active"
          <?php }  else { ?>
              "nav-link"
          <?php } ?>
          href="rules.php"><i class="fas fa-file-alt"></i> Règles
        </a>
      </li>

      <li class="nav-item">
        <a class=  <?   if($page=='credits'){ ?>
              "nav-link active"
          <?php    }  else {    ?>
              "nav-link"
          <?php    } ?>
          href="credits.php"><i class="fas fa-info-circle"></i> Crédits
        </a>
      </li>
    </ul>
    <?php
      if (isset($_SESSION['login'])) {
```

Navbar commence avec l'affichage de l'icône de page.

Ensuite une liste des pages en php qui récupère la variable $page vu dans les fichiers précédents.





```php
<?php
  if (isset($_SESSION['login'])) {
    echo '<ul class="account navbar-nav flex-fill justify-content-end align-items-center">
            <img class="img-fluid" src="https://picsum.photos/50">
            <li class="nav-item dropdown">
              <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"
              data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                '.ucfirst($_SESSION['login']).'
              </a>
              <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                <div class="d-flex flex-column">
                  <span class="w-100 dropdown-elements">Score: '.$_SESSION['score'].'</span>
                  <a href="logout.php" class="w-100 btn btn-logout btn-primary-outline">Me
                  déconnecter</a>
                </div>
              </div>
            </li>
          </ul>';
  }
  ?>
</div>
</nav>
```

Ensuite, si l'utilisateur est connecté on affiche son image, puis son nom. Si l'utilisateur clique sur cet affichage, on retrouve le menu déroulant avec le score de l'utilisateur et le bouton déconnexion.

Page score.php :

```php
<?php
  $page="";
  ini_set('display_errors', 1);
  include 'connexionbdd.php';
  session_start();
?>

<!DOCTYPE html>
<html lang="fr">
```

On a besoin de déclarer l'html pour réussir à l'ouvrir avec ajax. Ensuite on déclare les fonctions ajout_score et enleve_score. Ces deux fonctions récupèrent le score dans la BDD, pour ajoute, on y ajoute une valeur envoyée par javascript, pour enlever on y soustrait cette valeur. Ensuite on renvoi le résultat dans la BDD dans la variable score.

En dessous on récupère, si ajax a demandé l'ajout du score ou la diminution du score. Dans chaque cas on lance la fonction correspondante.

```php
<?php
//print_r($_SESSION['login']);
function ajout_score($score){

    include 'connexionbdd.php';
    //etape 1 recuperer le score
    $requete1 = $bdd->query("SELECT `score` FROM `utilisateur` WHERE `utilisateur`.
    `pseudo` = '".$_SESSION['login']."';");
    while ($data = $requete1->fetch())
  {
        $ancienScore = $data["score"];
    }
    $score += $ancienScore;
    $requete1->closeCursor();

    //etape 2 mettre a jour de score
    $requete2 = $bdd->query("UPDATE `utilisateur` SET `score` = '$score' WHERE
    `utilisateur`.`pseudo` = '".$_SESSION['login']."';");
    $requete2->closeCursor();

}

function enleve_score($score){
    include 'connexionbdd.php';
    //etape 1 recuperer le score
    $requete3 = $bdd->query("SELECT `score` FROM `utilisateur` WHERE `utilisateur`.
    `pseudo` = '".$_SESSION['login']."';");
    while ($data = $requete3->fetch())
  {
        $ancienScore = $data["score"];
    }
    $score = ($ancienScore-$score);
    $requete3->closeCursor();

    //etape 2 mettre a jour de score
    $requete4 = $bdd->query("UPDATE `utilisateur` SET `score` =  '$score'  WHERE
    `utilisateur`.`pseudo` = '".$_SESSION['login']."';");
    $requete4->closeCursor();

}
/*
if(isset($_GET["ajout_score"])) {ajout_score( $_GET["ajout_score"] );}
if(isset($_GET["enleve_score"])) {enleve_score( $_GET["enleve_score"] );}*/

if(isset($_POST["ajout_score"])) {ajout_score( $_POST["ajout_score"] );}
if(isset($_POST["enleve_score"])) {enleve_score( $_POST["enleve_score"] );}
```

Page script.js :

Sommaire :

```
/**
 * JS
 * table des functions :
 *
 * les sons play...()
 *
 * var getelementbyid
 *
 * ajaxAjoute()
 *
 * ajaxEnleve()
 *
 * retourAnimationChild23()
 *
 * resetRoulement()
 *
 * roulement()
 *
 * apresRoulement()
 *
 * initJouer3rouleaux()
 *
 * jeuToutNiveau()
 *
 * changementIconsExplic()
 *
 * choixNiveauBouton.addEventListener("click", function() {}
 *
 * remboursement()
 *
 *
 */
```

Choix du niveau :

```
/* init texte */
var choixNiveauOption = choixNiveau.value;
choixNiveauP.textContent = "Vous avez choisi le niveau "+ choixNiveauOption+ ".";

/* appel de function */
if (choixNiveauOption == 1){
    if(banditLvl1.className === 'banditContainerNone'){
        banditLvl1.className = 'banditContainerBlock';
    }
    choixNiveauP.textContent += "Une partie coute 10 clochettes";
    changementIconsExplic(choixNiveauOption);
    jeuToutNiveau(choix1Icons, manche1, animePieces1, banditLvl1, choixNiveauOption);
}
else if (choixNiveauOption == 2) {
    if(banditLvl2.className === 'banditContainerNone'){
        banditLvl2.className = 'banditContainerBlock';
    }
    choixNiveauP.textContent += "Une partie coute 20 clochettes";
    changementIconsExplic(choixNiveauOption);
    jeuToutNiveau(choix2Icons, manche2, animePieces2, banditLvl2, choixNiveauOption);
}
else if (choixNiveauOption == 3) {
```

On commence par la détection du choix du niveau, avant dernier élément du sommaire. Au choix du niveau, on affiche le choix dans la zone de texte prévu à cet effet, on change le container du niveau pour qu'il soit visible et on affiche le prix de la partie. Ensuite on lance la fonction changementIconesExplic car les niveaux 1 et 2 n'ont pas les mêmes icones que les niveaux 3 et 4.

Fonction jeuToutNiveau

```
/* ************************ fct jeuToutNiveau ************************ */
function jeuToutNiveau(choixIcon, manche, animePieces, banditLvl, niveau){
    /* SOMMAIRE de jeuToutNiveau()
    * init
    * texte explication
    * manche clique
    * choix icon clique
    * choix niveau clique
    */
```

La fonction jeu commence par un sommaire.

D'abord on initialise la fonction, on récupére les rouleaux du jeu, les 3 premiers puis le 4^{ieme} si le niveau sélectionné est 3 ou 4.

On rend visible l'explication des icônes et on rempli le texte en fonction du niveau. On défini un booléan « partie en cours ». Si aucune partie n'est en cours et que l'utilisateur appui sur le manche, on lance la partie. Si le joueur appui sur le bouton pour changer les icones, on vérifie si l'utilisateur peut payer la partie. S'il n'a pas assez de clochettes, un message est affiché et on quitte la boucle. Sinon, on enlève l'animation avant partie si elle est en cours, en utilisant également la fonction « initJouer3rouleaux » puis on vérifie si une partie est en cours.

Toujours si aucune partie n'est en cours, on change les icones des rouleaux et de l'explicaiton. Il y a deux boucle, icônes personnes vers fruits et fruits vers personnages. Ces deux boucles vérifient le niveau, pour 1 et 2 on modifie les rouleaux 1, 2 et 3, pour les niveaux 3 et 4 on modifie le rouleau 4 en plus. Pour la lecture du fichier on retrouve en commentaire « // end of … » à la fin des boucles.

```
boolPartieEnCours = true;
ajaxEnleve(prix,niveau);
playUnePiece();

/* roulement ! */
var randomR1 = Math.random() * (50 - 11) + 11; //min 1 tour
complet
var randomR2 = Math.random() * ((20 + randomR1) - randomR1) + randomR1;
//min les tours du r1
var randomR3 = Math.random() * ((15 + randomR2) - randomR2) + randomR2;
//min les tours du r2
var randomR4 = Math.random() * ((10 + randomR3) - randomR3) + randomR3;
//min les tours du r3
//on attend 0.3s avant de relancer (0.25s le temps de l'animation css)

/* bool d etat du roulement */
var roulementFini1 = false;
var roulementFini2 = false;
var roulementFini3 = false;
var roulementFini4 = false;
if(niveau == 1 || niveau == 2){ roulementFini4 = true; }
```

On met le booléen partie en cours en true, on diminue le score par le prix de la partie, on joue le son d'une piece puis on créer un nombre aléatoire du nombre de tour du rouleau 1, entre 50 et 11, 11 correspondant à un tour.

Puis les rouleaux suivant doivent au minimum faire le nombre de tour du rouleau précédent.

Ensuite on lance la fonction « roulement » autant de fois que le nombre aléatoire créé précédemment.

Pour le dernier tour du dernier rouleau, on lance la fonction « apresRoulement » puis on attend 2secondes avant de déclarer la partie en cours finie, donc pas de partie en cours.

Si une partie est en cours, on affiche un message d'erreur. A la fin de la fonction jeu, on détermine si l'utilisateur change de niveau pour remettre le bandit manchot en cours invisible. Pour cela on appel les fonctions « retourAnimationChild23 » et « resetRoulement ».

Fonction après roulement

Cette fonction vient finir la partie. On vérifie si les cases au centre sont les mêmes. Si oui, alors l'utilisateur a gagné, on affiche donc « Victoire ! » suivi d'un message indiquant la somme gagnée et on envoie cette somme dans la fonction « ajaxAjoute » pour mettre à jour la BDD. Sinon, on affiche « Perdu ! ». En cas d'erreur on affiche un message pour l'utilisateur et on rembourse le prix de la partie avec la fonction « remboursement ».

Fonction remboursement

En fonction du niveau de la partie, on envoie le prix de la partie dans la fonction « ajaxAjoute » pour rembourser l'utilisateur.

Fonctions ajax Ajout et Enleve

```
    $(document).ready(function(){
        $.ajax ({
            /*url : 'score.php?ajout_score='+ajout_score,
            type : 'GET',
            dataType : 'html',*/ //premier code fonctionnel, passage au POST pour la
            securite
            url : 'score.php',
            type: 'post',
            dataType : 'html',
            data: {'ajout_score': ajout_score},
            success : function(){
            },
            error : function(resultat, statut, erreur){
                remboursement(niveau);
                alert("Erreur");
            }
        });
    });
}
function ajaxEnleve(enleve_score, niveau){
    $(document).ready(function(){
        $.ajax ({
            /*url : 'score.php?enleve_score='+enleve_score,
            type : 'GET',*/ //pour la securite, POST
            url : 'score.php',
            type: 'post',
            dataType : 'html',
            data: {'enleve_score': enleve_score},
            success : function(){
            },
            error : function(resultat, statut, erreur){
                alert("Erreur");
                remboursement(niveau);
            }
        });
    });
```

On laisse en commentaire nos recherches car c'est la seule partie qui a vraiment posé un problème dans le code. Il s'agit d'une requête AJAX réalisée grâce aux framework jQuery.

Les deux fonctions utilisent $.ajax pour ouvrir le fichier « score.php ». En cas d'erreur lors de l'ouverture de score.php, on lance un remboursement. Sinon on envoi en POST pour la sécurité, soit la variable $_POST['ajout_score '] ou $_POST['enleve_score '].

Fonction retourAnimationChild23

Reset un bandit manchot pour lui remettre l'animation d'avant partie. Donc affecte ou enlève les classes CSS correspondantes.

Fonction resetRoulement

Enlève les classes CSS correspondants au roulement du bandit manchot.

Fonction roulement

En fonction de la case du rouleau qui a la classe « case_selected » qui correspond à la place du milieu, on affecte les class CSS au suivant, du bas vers le haut.

Fonction changementIconsExplic

Change les icones en fonction du niveau.

Fonction initJouer3rouleaux

```
/* init CSS */
/*r1*/
rouleau1.childNodes[1].classList.add('case_prev5');
rouleau1.childNodes[3].classList.add('case_prev4');
rouleau1.childNodes[5].classList.add('case_prev3');
rouleau1.childNodes[7].classList.add('case_prev2');
rouleau1.childNodes[9].classList.add('case_prev');
rouleau1.childNodes[11].classList.add('case_selected');
rouleau1.childNodes[13].classList.add('case_next');
rouleau1.childNodes[15].classList.add('case_next2');
rouleau1.childNodes[17].classList.add('case_next3');
rouleau1.childNodes[19].classList.add('case_next4');
rouleau1.childNodes[21].classList.add('case_next5');

/*r2*/
rouleau2.childNodes[1].classList.add('case_prev5');
rouleau2.childNodes[3].classList.add('case_prev4');
rouleau2.childNodes[5].classList.add('case_prev3');
rouleau2.childNodes[7].classList.add('case_prev2');
rouleau2.childNodes[9].classList.add('case_prev');
rouleau2.childNodes[11].classList.add('case_selected');
rouleau2.childNodes[13].classList.add('case_next');
rouleau2.childNodes[15].classList.add('case_next2');
rouleau2.childNodes[17].classList.add('case_next3');
rouleau2.childNodes[19].classList.add('case_next4');
rouleau2.childNodes[21].classList.add('case_next5');

/*r3*/
rouleau3.childNodes[1].classList.add('case_prev5');
rouleau3.childNodes[3].classList.add('case_prev4');
rouleau3.childNodes[5].classList.add('case_prev3');
rouleau3.childNodes[7].classList.add('case_prev2');
rouleau3.childNodes[9].classList.add('case_prev');
rouleau3.childNodes[11].classList.add('case_selected');
rouleau3.childNodes[13].classList.add('case_next');
rouleau3.childNodes[15].classList.add('case_next2');
rouleau3.childNodes[17].classList.add('case_next3');
rouleau3.childNodes[19].classList.add('case_next4');
rouleau3.childNodes[21].classList.add('case_next5');



/* parents */
rouleau1.classList.add('casesParent');
rouleau2.classList.add('casesParent');
rouleau3.classList.add('casesParent');
/ end of initJouer3rouleaux
```

On rend invisible les cases qui ne servent que pour l'animation d'avant partie. Ensuite on affecte les class CSS nécessaire pour le roulement. Soit, la case du centre, les 5 cases précédentes et les 5 cases suivantes. Visibles ci-contre.

Edit.php

```php
<?php
$page = 'edit';
ini_set('display_errors', 1);
include('connexionbdd.php');
session_start();
```

Comme la plupart des fichiers, édit commence avec la variable $page, la récupération de connxionbdd.php et de la session.

En fonction des modifications, le code va mettre à jours les données dans la base de données.