Manual del programador

DinoMathics

Autores:

Diana Edelmira Osorio Quiroga

Wilson David Garzón Duque

Fecha

04/02/2019

Tabla de contenido

1. Introducción al manual del programador 4
2. Base de datos 7
3. Lenguajes de programación usados 15
4. Haciendo la conexión con unity 15
5. Cargar datos 16

Tabla de Ilustraciones

Ilustración 1. Modelo relacional bases de datos 8
Ilustración 2. Conexión base de datos 10
Ilustración 3. Consultar Id del reto 11
Ilustración 4. Consultar Id usuario 11
Ilustración 5. Login 12
Ilustración 6. Datos retos resueltos 13
Ilustración 7. Nombre Personaje 13
Ilustración 8. Registrar usuario 14
Ilustración 9. Retos realizados 15
Ilustración 10. Coroutine para ingresar usuario 16

Introducción al manual del programador

El propósito de este manual es dar a conocer al lector las clases usadas para la creación del videojuego DinoMathics, se hará un breve repaso por los métodos empleados, las variables usadas y los scripts con el fin de dejar en claro el proceso de desarrollo del videojuego.

1. Scripts

Los scripts son archivos de texto que almacenan instrucciones de programación utilizando C++ como lenguaje de programación y visual Studio como editor. Los scripts creados en Unity son componentes de los objetos del proyecto. A continuación se explicaran los scripts generados en DinoMathics y su objetivo en el videojuego.

ArrojarComida.cs: Este Script maneja el comportamiento del nivel 2 “Alimenta a tu Dino” de la dificultad aprendiz, al oprimir la tecla espacio suelta una unidad de pollo la cual al colisionar con Dino cuenta 1 unidad.

Boton.cs: Verifica que la cantidad de alimento arrojado a Dino sea la cantidad que se pide en el contexto del reto.

Comida.cs: Maneja el comportamiento del alimento arrojado a Dino al ser arrojado con la tecla espacio.

ContandoEvento.cs: Cuando el alimento colisiona con Dino se agregara 1 unidad a la cuenta almacenada en pantalla.

BotonesA.cs: Muestra los paneles de ayuda, salir y créditos presentes en el menú principal.

LevelManager.cs: Este script se arrastra a un objeto vacío en el inspector de Unity, y dicho objeto vacío se arrastra a determinado botón, con el fin de vincular dicho botón a una escena o nivel en específico.

InicioManager.cs: En el menú de inicio verifica que el usuario desea jugar DinoMathics sin hacer uso de Bases de datos.

QuicesDBA.cs: Este script maneja los quices de sumas y restas, verificando la respuesta correcta y mostrando información de apoyo si la respuesta es incorrecta. También maneja el cambio entre escenas o retos en caso de que la respuesta sea correcta.

FollowCamPlayer.cs: En DinoMathics existen niveles en los cuales el personaje principal está en constante movimiento, este script se añade al personaje para que cuando este en movimiento la cámara principal se mueva con él y no se pierda de vista.

GameControlJumper.cs: Este script se implementó en los niveles que involucran la modalidad de juego “Salta con los números”, maneja los paneles de ayuda, tutorial, intenta de nuevo y de recompensa.

GameControlParallaxJumper.cs: Este script se implementó en los niveles que involucran la modalidad de juego “Salta con los números”, dichos niveles están ambientados en acantilados, por lo cual para agregarle más interactividad al reto, se implementó un efecto de movimiento, el cual consiste en que el fondo se mueva constantemente hacia una dirección, lo que genera la sensación de avance y movimiento. Este efecto se denomina “Parallax”.

PanelManagerJ.cs: Este script activa el panel de pausa, el cual contiene 3 opciones: continuar, ir a menú y reiniciar, además maneja el comportamiento de dichas opciones.

Player2.cs: Maneja el comportamiento de Dino en la modalidad de juego “Salta con los números”, consiste en hacer que Dino realice un salto cada que este haga contacto con el suelo o con determinadas plataformas.

SonidoManager.cs: Maneja los audios que relatan el objetivo de determinado reto.

GemaColision.cs: Este script se implementó en los niveles que involucran la modalidad de juego “Salta con los números”, cuando Dino llega a la meta y haga contacto con el trofeo allí ubicado, avanzara de reto o nivel.

Player.cs: Este script maneja el comportamiento de “AeroDino”, en la modalidad de juego “Vuela con los números”, le otorga movimiento en el eje X y Y, además de establecer los controles de movimiento y la colisión con los números.

Player1.cs: Este script maneja el comportamiento de “Dino”, en la modalidad de juego “Atrapa los números”, le otorga movimiento en el eje X , además de establecer los controles de movimiento, la colisión con los signos y los límites de movimiento.

Simbolos.cs: Este script se implementó en los niveles que involucran las modalidades de juego “Vuela con los números” y “Atrapa los números”, manejan el comportamiento de los números y símbolos que están cayendo constantemente en dichas modalidades de juego.

SpawnManager.cs: Este script maneja el tiempo de reaparición y la ubicación de los objetos que están cayendo de manera constante en las modalidades de juego “Vuela con los números” y “Atrapa los números”.

2. Base de datos

Para la creación de la base de datos, hicimos uso de 4 tablas

Usuario: Esta tabla contiene un id, el nombre de usuario y el personaje elegido al momento de realizar el registro.

Reto: En esta tabla se almacenan los datos correspondientes a todos los retos que existen en el videojuego, contiene un nombre del reto, el logro obtenido cuando se resuelve, y un id.

Logro: la tabla logro es la tabla encargada de almacenar los logros que serán otorgados cada vez que el usuario resuelva un reto, esta tabla contiene; nombre del reto y un id.

Usuario_reto: contiene los retos que han sido resueltos por el usuario, allí se almacena

Las bases de datos se crearon bajo la siguiente estructura

```
CREATE TABLE USUARIO (
   id_user INT UNSIGNED AUTO_INCREMENT,
   nombre VARCHAR(30) NOT NULL,
   personaje VARCHAR(30),
   PRIMARY KEY(id_user)
);
CREATE TABLE LOGRO (
   id_log INT UNSIGNED AUTO_INCREMENT,
   nombre_log VARCHAR(30) NOT NULL,
   PRIMARY KEY(id_log)
);

CREATE TABLE RETO (
   id_reto INT UNSIGNED AUTO_INCREMENT,
   nombre_re VARCHAR(30) NOT NULL,,
   id_log INT UNSIGNED NOT NULL,
   PRIMARY KEY(id_reto),
   INDEX fk_id_log (id_log),
   CONSTRAINT fk_id_log FOREIGN KEY (id_log) REFERENCES
LOGRO(id_log)
);
```

```
CREATE TABLE USUARIO_RETO (
   id_resu INT NOT NULL AUTO_INCREMENT,
   tiempo TIME NOT NULL,
   n_intentos INT NOT NULL,
   id_reto INT UNSIGNED NOT NULL,
   id_user INT UNSIGNED NOT NULL,
   PRIMARY KEY (id_resu),
   puntaje INT NOT NULL,
   INDEX fk_id_reto (id_reto),
   INDEX fk_id_user (id_user),
   CONSTRAINT fk_id_reto FOREIGN KEY (id_reto) REFERENCES RETO
(id_reto),
   CONSTRAINT fk_id_user FOREIGN KEY (id_user) REFERENCES USUARIO
(id_user)
);
```

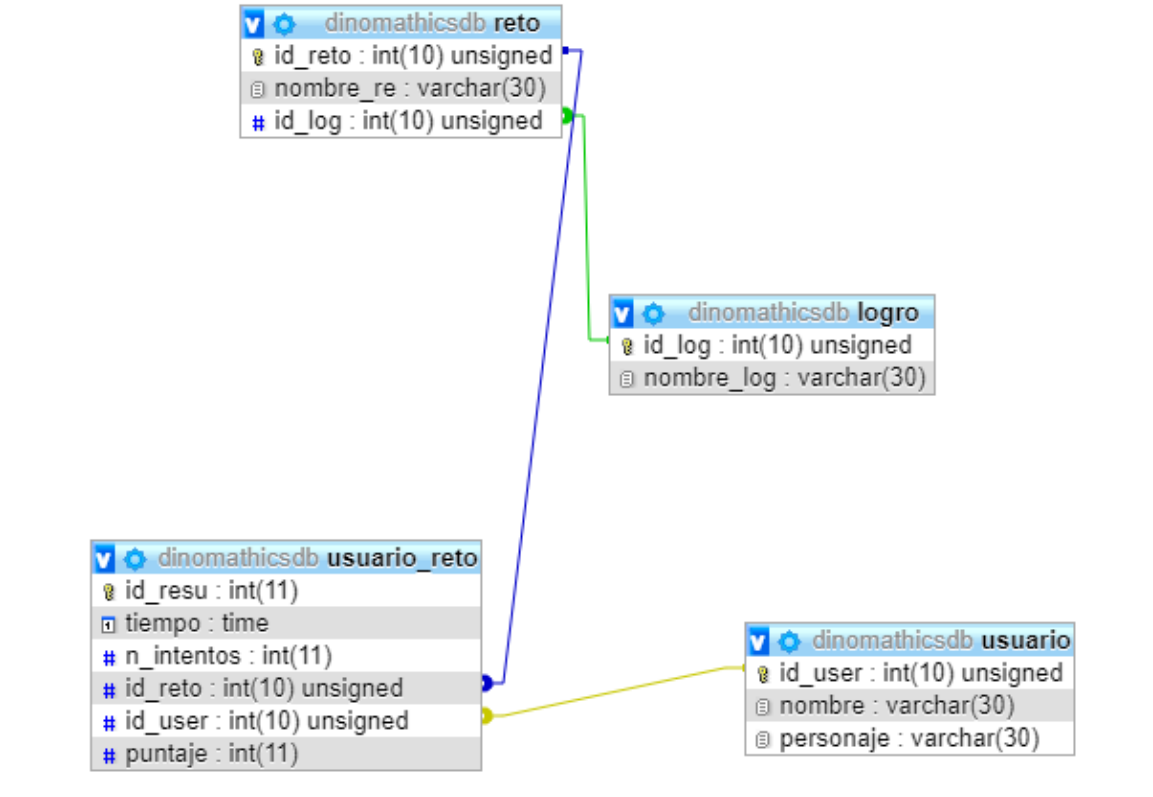


Ilustración 1. Modelo relacional bases de datos

Las inserciones a las tablas logro y a la tabla reto, fueron de la siguiente manera:

```
INSERT INTO LOGRO VALUES (null,'Gema');
INSERT INTO LOGRO VALUES (null,'Trofeo');
```

```
INSERT INTO RETO VALUES (null,'jumper 2 en 2',2);
INSERT INTO RETO VALUES (null,'jumper 1 en 1',2);
INSERT INTO RETO VALUES (null,'jumper 3 en 3',2);
INSERT INTO RETO VALUES (null,'jumper 5 en 5',2);
INSERT INTO RETO VALUES (null,'Quiz jumper',1);
INSERT INTO RETO VALUES (null,'Atrapar signo +',2);
INSERT INTO RETO VALUES (null,'Atrapar signo -',2);
INSERT INTO RETO VALUES (null,'Atrapar signo =',2);
INSERT INTO RETO VALUES (null,'Quiz signos',1);
INSERT INTO RETO VALUES (null,'Atrapar numero 3',2);
INSERT INTO RETO VALUES (null,'Atrapar numero 5',2);
INSERT INTO RETO VALUES (null,'Atrapar numero 6',2);
INSERT INTO RETO VALUES (null,'Atrapar numero 8',2);
INSERT INTO RETO VALUES (null,'Atrapar numero 9',2);
INSERT INTO RETO VALUES (null,'Quiz Atrapar numeros',1);
INSERT INTO RETO VALUES (null,'Quiz suma huesos',1);
INSERT INTO RETO VALUES (null,'Quiz resta huesos',1);
INSERT INTO RETO VALUES (null,'Quiz completar suma',2);
INSERT INTO RETO VALUES (null,'Quiz completar resta',2);
INSERT INTO RETO VALUES (null,'Quiz suma serializar',2);
INSERT INTO RETO VALUES (null,'Quiz resta serializar',2);
INSERT INTO RETO VALUES (null,'Arrastrar objetos al contenedor',2);
INSERT INTO RETO VALUES (null,'Quiz arrastrar objetos',1);
INSERT INTO RETO VALUES (null,'Ubicar objetos(izq, der, arriba, abajo,
dentro)',1);
INSERT INTO RETO VALUES (null,'Arrastrar objetos de medicion(regla)',1);
INSERT INTO RETO VALUES (null,'Arrastrar medidas jarras',2);
INSERT INTO RETO VALUES (null,'Medir objetos(lapiz)',1);
INSERT INTO RETO VALUES (null,'Quiz suma huevos',2);
INSERT INTO RETO VALUES (null,'Quiz resta huevos',2);
```

2.1 Conexión mediante php

Para realizar la conexión y posteriormente consultas e inserción de datos, usamos php, se hizo uso de los siguientes 6 archivos.

- **Conectar.php**

  Obtenemos los valores necesarios para realizar la conexión, dependiendo de la configuración del servidor local.



Ilustración 2. Conexión base de datos

- **ConsultarIdReto.php**

  Este archivo es usado para realizar la consulta a la tabla reto, y conocer el id_reto, del reto en particular que haya superado el usuario. si logra encontrar el reto procederá a imprimirlo, pero si la búsqueda no arroja ningún resultado imprime un mensaje 402.

```php
<?php

   include('Conectar.php');

   $reto     = $_GET['reto'];
   if(!$conexion){
      echo ("Fallo conexion");
   }
   else{
      $query = "SELECT id_reto FROM reto WHERE nombre_re LIKE '$reto'";
      $resultado = mysqli_query($conexion, $query);
      if(mysqli_num_rows($resultado)>0){
         while ($row = mysqli_fetch_assoc($resultado)) {
            echo $row['id_reto'];
         }
         //echo "201";
      }else if(mysqli_num_rows($resultado)==0){

         echo "402";
      }
   }

?>
```

Ilustración 3. Consultar Id del reto

- **ConsultarIdUsuario.php**

  Con este archivo realizamos la consulta a la tabla usuario (en todas las consultas realizadas obtenemos los datos a través del método $_GET) a través de la variable $nombre y luego se imprime el resultado de la variable. Si no logra encontrar el usuario en la tabla imprimirá un mensaje 402.

```php
<?php

   include('Conectar.php');

   $nombre     = $_GET['nombre'];
   if(!$conexion){
      echo ("Fallo conexion");
   }
   else{
      $query = "SELECT id_user FROM usuario WHERE nombre LIKE '$nombre'";
      $resultado = mysqli_query($conexion, $query);
      if(mysqli_num_rows($resultado)>0){
         while ($row = mysqli_fetch_assoc($resultado)) {
            echo $row['id_user'];
         }
         //echo "201";
      }else if(mysqli_num_rows($resultado)==0){

         echo "402";
      }
   }

?>
```

Ilustración 4. Consultar Id usuario

- **Login.php**

  El archivo del login solo usa una variable para obtener por el método $_GET el nombre y hacer la verificación de la existencia del usuario.

```php
<?php

   include('Conectar.php');

   $nombre = $_GET['nombre'];

   if(!$conexion){
      echo ("Fallo conexion");
   }
   else{
      $query = "SELECT * FROM usuario WHERE nombre LIKE '$nombre'";
      $resultado = mysqli_query($conexion, $query);
      if(mysqli_num_rows($resultado)>0){
         echo "200";
      }else{
         echo "401";
      }
   }

?>
```

Ilustración 5. Login

- **MostrarDatos.php**

  Este archivo es el encargado de mostrar todo el avance del jugador que se pida, solo obtenemos el nombre por el método $_GET, y realizamos la unión de las tablas necesarias para imprimir toda la información. En caso de que el jugador no se encuentre en la base de datos, enviara un mensaje 401.

```php
<?php

   include('Conectar.php');

   $nombre     = $_GET['nombre'];
   if(!$conexion){
      echo ("Fallo conexion");
   }
   else{
         //echo "si lo conecteee";
      $query = "SELECT nombre, nombre_re, nombre_log, tiempo, n_intentos, puntaje FROM usuario
      join usuario_reto on usuario_reto.id_user=usuario.id_user
      join reto on  usuario_reto.id_reto = reto.id_reto
      join logro on logro.id_log = reto.id_log WHERE nombre = '$nombre'";
      $resultado = mysqli_query($conexion, $query);
      if(mysqli_num_rows($resultado)>0){
         while ($row = mysqli_fetch_assoc($resultado)) {
            echo $row['nombre']."|".$row['nombre_re']."|".
            $row['nombre_log']."|".$row['tiempo']."|".
            $row['n_intentos']."|".$row['puntaje']."|";
         }
         //echo "201";
      }else if(mysqli_num_rows($resultado)==0){

         echo "401";
      }
   }

?>
```

Ilustración 6. Datos retos resueltos

- **NombreDino.php**

En esta clase consultamos el nombre del personaje elegido por el usuario, es decir para verificar si escogió al personaje Dino o a Dina. En caso de no encontrar el usuario registrado en la base de datos enviara un mensaje 402.

```php
<?php

   include('Conectar.php');

   $nombre     = $_GET['nombre'];
   if(!$conexion){
      echo ("Fallo conexion");
   }
   else{
         //echo "si lo conecteee";
      $query = "SELECT personaje FROM usuario WHERE nombre LIKE '$nombre'";
      $resultado = mysqli_query($conexion, $query);
      if(mysqli_num_rows($resultado)>0){
         while ($row = mysqli_fetch_assoc($resultado)) {
            echo $row['personaje'];
         }
         //echo "201";
      }else if(mysqli_num_rows($resultado)==0){

         echo "402";
      }
   }

?>
```

Ilustración 7. Nombre Personaje

- **Registrar.php**

Insertamos los datos de usuario a la tabla usuario de la base de datos. Obtenemos el nombre del usuario y el personaje por el método $_GET. Se hace la verificación que el nombre de usuario no se encuentre registrado, si el nombre está registrado envía un mensaje 402 que indica que no se puede hacer la inserción de datos. Si la inserción de datos culmina satisfactoriamente, se envía un mensaje 201 confirmado el proceso.

```php
<?php

   include('Conectar.php');

   $nombre = $_GET['nombre'];
   $nombreDino = $_GET['dino'];
   if(!$conexion){
      echo ("Fallo conexion");
   }
   else{
      if (isset($_GET['nombre'], $_GET['dino'])) {
         if (!empty($nombreDino) and !empty($nombre)) {
            $query = "SELECT * FROM usuario WHERE nombre LIKE '$nombre'";
            $resultado = mysqli_query($conexion, $query);
            if(mysqli_num_rows($resultado)>0){
                echo "402";
            }else if(mysqli_num_rows($resultado)==0){
                $query = "INSERT INTO usuario VALUES(NULL,'$nombre','$nombreDino')";
                $resultado = mysqli_query($conexion, $query);
                echo "201";
            }
         }else{
            echo "Campos vacios";
         }

      }else
      {
         echo "Faltan campos";
      }

   }

?>
```

Ilustración 8. Registrar usuario

- **UsuarioReto.php**

En esta clase hacemos la inserción de los datos a la tabla usuario_reto, cada que un usuario en Unity complete un reto, se usara este script para almacenar los datos en la tabla, en caso de que el reto ya haya sido realizado por el usuario, imprimirá un mensaje 402, que indicara que el reto ya fue resuelto por ese usuario y no será insertado, si el usuario resuelve un reto por primera vez el archivo enviara un mensaje 201 mostrando que la operación fue correcta.

```php
<?php

   include('Conectar.php');

   $tiempo     = $_GET['tiempo'];
   $n_intentos = $_GET['nIntentos'];
   $id_reto    = $_GET['reto'];
   $id_user    = $_GET['idUser'];
   $puntaje    = $_GET['puntaje'];
   if(!$conexion){
      echo ("Fallo conexion");
   }
   else{
      $query = "SELECT * FROM usuario_reto WHERE id_user LIKE '$id_user'AND id_reto LIKE '$id_reto'";
      $resultado = mysqli_query($conexion, $query);
      if(mysqli_num_rows($resultado)>0){
         echo "402";
      }else if(mysqli_num_rows($resultado)==0){
         $query = "INSERT INTO usuario_reto VALUES(NULL,'$tiempo','$n_intentos', '$id_reto', '$id_user','$puntaje')";

         $resultado = mysqli_query($conexion, $query);
         echo "201";
      }
   }

?>
```

Ilustración 9. Retos realizados

3. Lenguajes de programación usados

El lenguaje principal que se usó durante el proyecto fue c#, con el creamos las clases que sirvieron para crear las escenas del videojuego.

Para la conexión con la base de datos se usó el lenguaje php, con php hicimos la conexión al servidor y base de datos, realizamos las consultas y definimos el método para obtener los datos ingresados a la consulta.

Para crear la estructura de la base de datos utilizamos Mysql, con Mysql creamos las tablas e ingresamos los datos a cada una de ellas.

Haciendo la conexión con Unity

Para el manejo de bases de datos en Mysql Unity ofrece diferentes métodos, en todos los métodos ofrecidos no es necesario usar php, en algunos se puede usar archivos dll, pero restringen la cantidad de consultas a la base de datos. Para poder usar bases de datos en Mysql y usar php, Unity implementa el método WWW con este método podemos enviar la url de donde se encuentran alojados los archivos php que realizan las acciones a la base de datos.

Como cuando usamos php para realizar todas las consultas e inserciones definimos que los datos necesarios serian traídos por el método $_GET, incluimos

directamente en la url el parámetro que solicita los datos. Para usar el modulo WWW, es necesario hacer uso de Coroutines. Las coroutines, permiten que la ejecución del código se pueda reanudar o detener en cada frame.

```
55
56
57      IEnumerator Ingresar()
58      {
59          WWW conn = new WWW("http://localhost/DinoMathicsPHP/login.php?nombre="+nombre.text);
60          yield return (conn);
61          Debug.Log(conn.text);
62          if(conn.text == "200")
63          {
64              Guardar();
65              loadMenuPrincipal();
66              print("Sesion establecida");
67          }
68          else if(conn.text == "401")
69          {
70              error.SetActive(true);
71              nombre.text = "";
72              print("Usuario no existe");
73          }else
74          {
75              errorBD.SetActive(true);
76              nombre.text = "";
77              print("No se pudo conectar con la base de datos");
78          }
79      }
```

Ilustración 10. Coroutine para ingresar usuario

En la ilustración anterior, usamos una coroutine para crear el ingreso de los usuarios, se hace uso del módulo WWW al cual le pasamos el url donde está ubicado el archivo login.php, y le enviamos como parámetro una variable text.

Para iniciar la coroutina es necesario hacer uso del método StartCoroutine();

El método que inicia la coroutine es puesto dentro de otro método, para posteriormente ser ejecutado cuando se active un evento definido.

Cargar datos

Para cargar los datos de un jugador y/o verificar la existencia en la base de datos, usamos un módulo de Unity llamado playerprefbs, este módulo nos permite guardar los datos de una variable y asociarlos a una clave, para de esta manera poder usarlo en cualquier escena del juego sin que los datos sean borrados. Con ayuda de este módulo obteníamos el nombre del jugador en cada escena del videojuego y de esta manera podíamos hacer la inserciones a la base de datos de una manera más simple.

Para guardar el nombre de un jugador usando playerprefbs usamos la siguiente sintaxis.

PlayerPrefs.SetString ("Jugador", nombre.text);

SetString permite almacenar un valor de tipo string.

nombre.text contiene el nombre del jugador introducido en un inputfield. La clave generada para asociar al nombre del usuario es, “Jugador”.

Para obtener los datos y guardarlos en una variable usamos la siguiente sintaxis.

nombrePlayer = PlayerPrefs.GetString("jugador", nombre.text);

Así la variable nombrePlayer que contiene el nombre del jugador puede ser usada en cualquier escena del videojuego sin que su valor sea eliminado.

Creando una instancia

El objeto PlayerPrebs se encuentra dentro de la clase login.cs, para usar este podemos hacer una instancia en cualquier escena, por ejemplo para hacer una consulta en para buscar el id de un usuario, lo hacemos dentro de la clase RetoResuelto.cs y creamos una instancia del objeto PayerPrefbs.

private Login player;

player = GameObject.FindObjectOfType<Login>();

Con el objeto player podemos acceder a la variable donde se encuentra el nombre del jugador para realizar la consulta.

```
82
83  IEnumerator ConsultarIdUsuario()
84  {
85          WWW conn = new WWW("http://localhost/DinoMathicsPHP/ConsultarIdUsuario.php?nombre=" + player.nombrePlayer);
86          yield return (conn);
87          Debug.Log(conn.text);
88          if (conn.text == "401")
89          {
90              print("Usuario incorrecto");
91          }
92          else
93          {
94              idUser = int.Parse(conn.text);
95
96              Debug.Log(idUser);
97          }
98  }
99
```

Ilustración 11. Consultar Id usuario usando playerPrefbs