Casse Briques – Dernière partie

1) On doit perdre quand on rate la balle !
On supposera que le joueur perd quand la balle descend plus bas que la raquette.

il faut vérifier que le point le plus bas de la balle (Shape1.Top + Shape1.Height) est plus grand que raquette.Top + raquette.Height

Ensuite on affiche un message (perdu), on replace la balle au milieu de la page et on recommence.

Comme il commence à y avoir pas mal d code, la notation …/… indique qu'un a pas copié tout le code

Dans le fichie de code , cherche la procédure nommée

```
procedure TForm1.Timer1Timer(Sender: TObject);
  .../...
  Shape1.Top:=Shape1.Top+dy;
  Shape1.Left:=Shape1.Left+dx;
  Raquette.Left:=Raquette.Left+pas;
  pas:=0;
   if(Shape1.Top+Shape1.height > Raquette.Top + Raquette.Height) then
      begin
       showmessage('Perdu');
       dy:=5;
       //on place la balle au premier quart de l'écran en hauteur
       // et à la moitié en largeur
       Shape1.Top:=Round( 1*Form1.Top/4  );
       Shape1.Left:= Round(Form1.Width/2);
      end;
end;
```

2) Les briques , les briques ……. bon d'accord
On va créer un tableau de briques, cela sera plus simple pour la suite .

D'abord déclarer la variable tableau qui contiendra les briques, et ensuite préparer la procédure qui va créer le tableau de briques, ajouter les lignes en gras

Vers la ligne 30 du code,

```
 procedure Timer1Timer(Sender: TObject);
  private
    { private declarations }
```

```
    FBlocks : array of TShape; //declare le tableau de briques
     procedure CreeMur(Sender: TObject);
  public
    { public declarations }
  end;
```

Maintenant il faut écrire les codes correspondants , plus bans dans le code (Vers la ligne 103), juste au dessus de la procédure TForm1.MenuItem2Click

```
procedure TForm1.CreeMur(Sender:Tobject);
begin
    SetLength(FBlocks, 10);
     FBlocks[0]:=TShape.Create(Application);
     FBlocks[0].Parent:=Form1;
     FBlocks[0].Shape:=stRectangle;
     FBlocks[0].Height:=30;
     FBlocks[0].Width:=round(Form1.Width/10);
     FBlocks[0].Left:=0;
     FBlocks[0].Top:=50;
end;

procedure TForm1.MenuItem2Click(Sender: TObject);
```

dans CreeMur, que fait-on ?
on dit que le mur contiendra 10 briques , pour tester , on ne définit que la première brique , la ligne FBlocks[0].Width:=round(Form1.Width/10), détermine la largeur de la brique comme étant 1/10 de la largeur du formulaire (la fenêtre de jeu pour nous).

Maintenant que nous avons défini la procedure CreeMur, il faut l'utiliser , dabs la procedure : (ligne 122)

```
procedure TForm1.MenuItem2Click(Sender: TObject);
var
  i:integer;
begin
    Shape1.Width:=10;
    Shape1.Height:=10;
    Timer1.Enabled:=True;
     CreeMur(Sender);
end;
```

On reprend la procedure
procedure TForm1.CreeMur(Sender:Tobject);

on fait une boucle (boucle for) pour créer 10 briques, attention la première brique sera nommée FBlocks[0], et la dernière FBlocks[9].

```
procedure TForm1.CreeMur(Sender:Tobject);
var
  I :integer;
begin
  SetLength(FBlocks, 10);
  For I:=0 To 9 do
      begin
          FBlocks[I]:=TShape.Create(Application);
          FBlocks[I].Parent:=Form1;
          FBlocks[I].Shape:=stRectangle;
          FBlocks[I].Height:=30;
          FBlocks[I].Width:=round(Form1.Width/10);
          FBlocks[I].Left:=I*FBlocks[I].Width;
          FBlocks[I].Top:=0;

      end;
end;
```

On enregistre et on test … trop fort. Bon la balle traverse un peu les briques mais sinon c'est bien.
On va, pour chaque brique tester si elle est en contact avec la balle . Ce test doit se faire à chaque animation, donc dans la procedure TForm1.Timer1Timer.

```
 Shape1.Top:=Round( 1*Form1.Top/4  );
     Shape1.Left:= Round(Form1.Width/2);
    end;
  For I:=0 To 9 do
      begin
          If((Shape1.Top -  (FBlocks[I].Top+ FBlocks[I].Height ) <5)
             and (Shape1.Left >  FBlocks[I].Left)
             and (Shape1.Left+Shape1.Width <  FBlocks[I].Left +
FBlocks[I].Width))
          then dy:=-dy  //on change le sens de la balle
      end;
end;
```

procedure TForm1.CreeMur(Sender:Tobject);

On teste …. OK . Mais la brique reste en place, on va utiliser la propriété visible de chaque brique pour les faire disparaître, cependant, il faudra en tenir compte dans le test …

Modifier la ligne (ap eu ès 108) ainsi
then **begin** dy:=-dy; **FBlocks[I].Visible:=False; end;** //on change le sens de la balle

Problème, si la brique n'est plus présente, alors, il ne faut pas faire le test de collision ! (Ajouter le ligne en gras)

```
For I:=0 To 9 do
    begin
        if(FBlocks[I].Visible = True) then
        If((Shape1.Top -  (FBlocks[I].Top+ FBlocks[I].Height ) <5)
          and (Shape1.Left >  FBlocks[I].Left)
          and (Shape1.Left+Shape1.Width <  FBlocks[I].Left + FBlocks[I].Width))
        then begin dy:=-dy; FBlocks[I].Visible:=False; end; //on change le sens de la
balle
    end;
```

A part les tsts de collision qui sont à améliorer, cela fonctionne pas mal, il faut maintenant détecter le fin de partie (plus de brique).

On déclare une variable Fin de type boolean , que l'on place a vrai, si une brique est encore visible, alors on bascule Fin à False et on arrête pas la partie .

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
  I :integer;
   Fin: Boolean;
begin
```

puis plus bas :

```
For I:=0 To 9 do
     begin
         if(FBlocks[I].Visible = True) then
           begin
               Fin :=False;
               If((Shape1.Top -  (FBlocks[I].Top+ FBlocks[I].Height ) <5)
               and (Shape1.Left >  FBlocks[I].Left)
               and (Shape1.Left+Shape1.Width <  FBlocks[I].Left + FBlocks[I].Width))
               then begin dy:=-dy; FBlocks[I].Visible:=False; end; //on change le sens de
la balle
         end;
     end;
  if  (Fin) then showmessage('Bravo');
```

Voola, il ne reste plus qu'a ajouter vos petites idées (balle qui accélère sur une brique spéciale ...