Zmienne, nazwy metod i stringi z dotyczące liczby żyć zawierają „Lives” - to jest czasownik, powinno być Lifes 😊

ViewModel.LoadFont brakuje using na GetManifestResourceStream

private List<Brick> bricksToDraw = new List<Brick>();

public List<Brick> Bricks => bricksToDraw;

Nie potrzebujesz pola bricksToDraw. Można po prostu zrobić

public List<Brick> Bricks {get;} = new List<Brick>();

ViewModel.PrepareLivesAndScoresMessage()

Vector2f rect = new Vector2f(offsetX, (float)height - localBounds.Height - offsetY);

localBounds.Height jest typu float więc nie potrzeba rzutować height na float

message.Color = new Color(Color.White);

Tworzenie nowego Color nie jest potrzebne

message.Color = Color.White;

name = name.Replace("/", ".");

name = "zbrozonoidAssets.Fonts." + name;

Jedna linijka i czytelniej

string fullName = $"zbrozonoidAssets.Fonts.{name.Replace("/", ".")}";

ViewModel LoadFont i LoadBackground zawierają tę samą logikę.

Można by to wyciągnąć do osobnej metody np. coś w stylu:

private T LoadResource<T>(string name)

{

AssemblyName assemblyName = new AssemblyName(@"zbrozonoidAssets");

Assembly assembly = Assembly.Load(assemblyName);

using (Stream resourceStream = assembly.GetManifestResourceStream(name))

{

if (resourceStream == null)

{

return default(T);

}

return (T)Activator.CreateInstance(typeof(T), resourceStream);

}

}

private Font LoadFont(string name)

{

return LoadResource<Font>($"zbrozonoidAssets.Fonts.{name.Replace("/", ".")}");

}

Metody Prepare…Message też mają powtarzająca się logikę. Można by zrobić jedną metodą ShowMessage która dostawałaby tekst, rozmiar, kolor i coś oznaczającego pozycję na ekranie np. jakiś enum w stylu Center, TopRight. Tak żeby ta logika pobierająca rozmiar ekranu i ustawiająca pozycję była tylko w jednym miejscu.

ViewModel Font i Background mają publiczne sety, ale nic ich nie ustawia z zewnątrz

if (currentState != null)

{

currentState.Action();

}

Można uprościć do:

currentState?.Action();

private void DrawTail(IBall ball)

{

ITail tail = game.TailManager.Find(ball);

if (tail != null)

{

foreach (Position position in tail)

{

…

ball.GetSize(out int width, out int height);

…

}

}

}

Rozmiar ball się nie zmienia więc można zrobić GetSize raz – przed pętlą foreach.