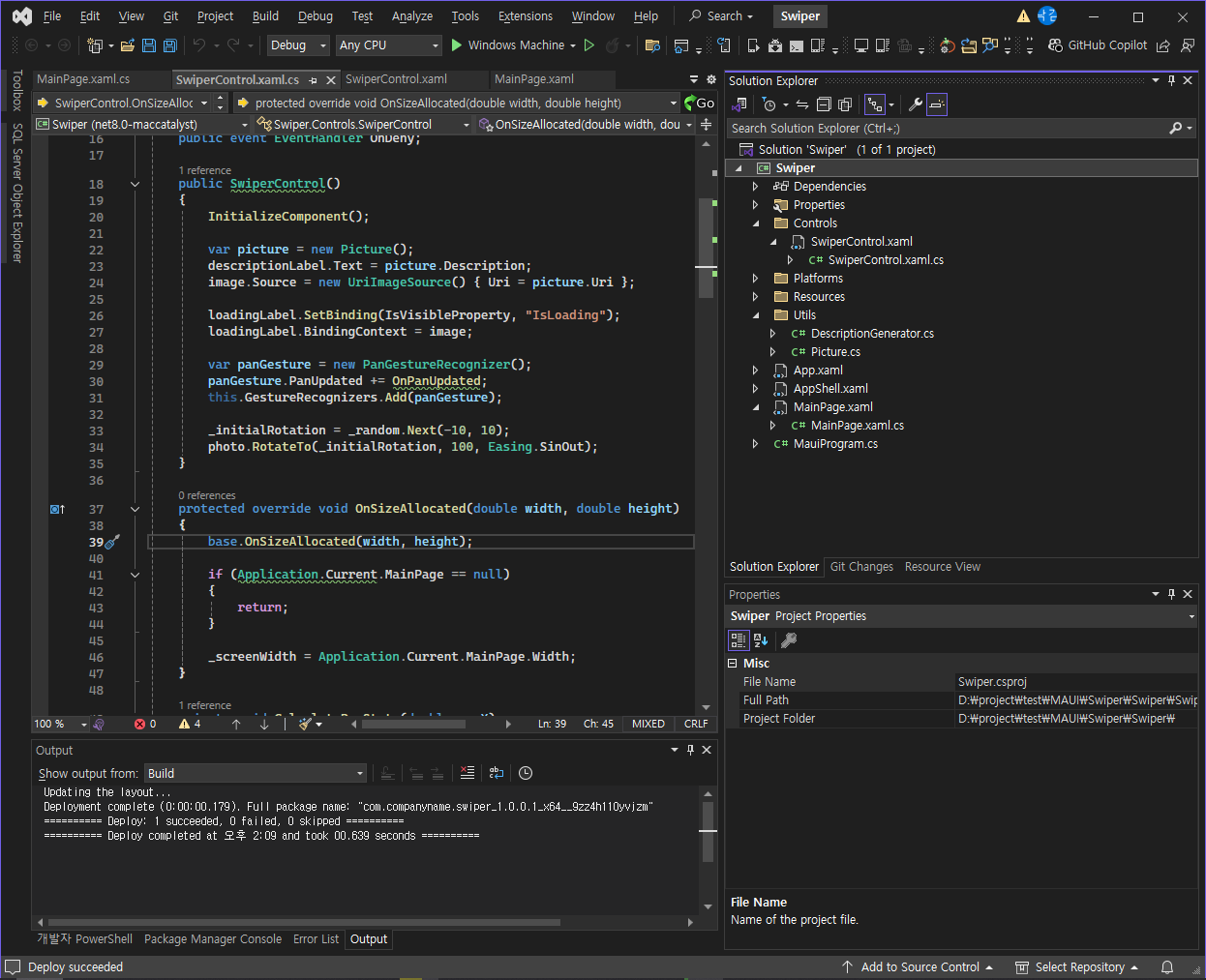
MAUI – SWIPER



Controls/SwiperControl.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentView xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="Swiper.Controls.SwiperControl">  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="100" />  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="100" />  </Grid.ColumnDefinitions>  <!-- ContentView for photo here -->  <ContentView x:Name="photo" Padding="40" Grid.ColumnSpan="3" >  <Grid x:Name="photoGrid" BackgroundColor="Black" Padding="1" >  <Grid.RowDefinitions>  <RowDefinition Height="\*" />  <RowDefinition Height="40" />  </Grid.RowDefinitions>  <BoxView Grid.RowSpan="2" Color="White" />  <Image x:Name="image" Margin="10" BackgroundColor="#AAAAAA" Aspect="AspectFill" />  <Label x:Name="loadingLabel" Text="Loading..." TextColor="White" FontSize="Large" FontAttributes="Bold" HorizontalOptions="Center" VerticalOptions="Center" />  <Label Grid.Row="1" x:Name="descriptionLabel" Margin="10,0" Text="A picture of grandpa" FontFamily="Bradley Hand" />  </Grid>  </ContentView>  <!-- StackLayout for like here -->  <StackLayout x:Name="likeStackLayout" Grid.Column="2" Opacity="0" Padding="0, 100" HorizontalOptions="Center">  <Label Text="LIKE" TextColor="Lime" FontSize="30" Rotation="30" FontAttributes="Bold" />  </StackLayout>  <!-- StackLayout for deny here -->  <StackLayout x:Name="denyStackLayout" Opacity="0" Padding="0, 100" HorizontalOptions="Center">  <Label Text="DENY" TextColor="Red" FontSize="30" Rotation="-20" FontAttributes="Bold" />  </StackLayout>  </Grid>  </ContentView> |

Controls/SwiperControl.xaml.cs

|  |
| --- |
| using Swiper.Utils;  namespace Swiper.Controls;  public partial class SwiperControl : ContentView  {  private readonly double \_initialRotation;  private static readonly Random \_random = new Random();  private double \_screenWidth = -1;  private const double DeadZone = 0.4d;  private const double DecisionThreshold = 0.4d;  public event EventHandler OnLike;  public event EventHandler OnDeny;  public SwiperControl()  {  InitializeComponent();  var picture = new Picture();  descriptionLabel.Text = picture.Description;  image.Source = new UriImageSource() { Uri = picture.Uri };  loadingLabel.SetBinding(IsVisibleProperty, "IsLoading");  loadingLabel.BindingContext = image;  var panGesture = new PanGestureRecognizer();  panGesture.PanUpdated += OnPanUpdated;  this.GestureRecognizers.Add(panGesture);  \_initialRotation = \_random.Next(-10, 10);  photo.RotateTo(\_initialRotation, 100, Easing.SinOut);  }  protected override void OnSizeAllocated(double width, double height)  {  base.OnSizeAllocated(width, height);  if (Application.Current.MainPage == null)  {  return;  }  \_screenWidth = Application.Current.MainPage.Width;  }  private void CalculatePanState(double panX)  {  var width = \_screenWidth == -1 ? 400 : \_screenWidth;  var halfScreenWidth = width / 2;  var deadZoneEnd = DeadZone \* halfScreenWidth;  if (Math.Abs(panX) < deadZoneEnd)  {  return;  }  var passedDeadzone = panX < 0 ? panX + deadZoneEnd : panX - deadZoneEnd;  var decisionZoneEnd = DecisionThreshold \* halfScreenWidth;  var opacity = passedDeadzone / decisionZoneEnd;  opacity = double.Clamp(opacity, -1d, 1d);  likeStackLayout.Opacity = opacity;  denyStackLayout.Opacity = -opacity;  }  private bool CheckForExitCriteria()  {  var width = \_screenWidth == -1 ? 400 : \_screenWidth;  var halfScreenWidth = width / 2;  var decisionBreakpoint = DeadZone \* halfScreenWidth;  return (Math.Abs(photo.TranslationX) > decisionBreakpoint);  }  private void Exit()  {  MainThread.BeginInvokeOnMainThread(async () =>  {  var direction = photo.TranslationX < 0 ? -1 : 1;  if (direction > 0)  {  OnLike?.Invoke(this, new EventArgs());  }  if (direction < 0)  {  OnDeny?.Invoke(this, new EventArgs());  }  await photo.TranslateTo(photo.TranslationX + (\_screenWidth \* direction), photo.TranslationY, 200, Easing.CubicIn);  var parent = Parent as Layout;  parent?.Children.Remove(this);  });  }  private void OnPanUpdated(object sender, PanUpdatedEventArgs e)  {  switch (e.StatusType)  {  case GestureStatus.Started:  PanStarted();  break;  case GestureStatus.Running:  PanRunning(e);  break;  case GestureStatus.Completed:  PanCompleted();  break;  }  }  private void PanStarted()  {  photo.ScaleTo(1.1, 100);  }  private void PanRunning(PanUpdatedEventArgs e)  {  photo.TranslationX = e.TotalX;  photo.TranslationY = e.TotalY;  photo.Rotation = \_initialRotation + (photo.TranslationX / 25);  CalculatePanState(e.TotalX);  }  private void PanCompleted()  {  if (CheckForExitCriteria())  {  Exit();  }  likeStackLayout.Opacity = 0;  denyStackLayout.Opacity = 0;  photo.TranslateTo(0, 0, 250, Easing.SpringOut);  photo.RotateTo(\_initialRotation, 250, Easing.SpringOut);  photo.ScaleTo(1, 250);  }  } |

Utils/DescriptionGenerator.cs

|  |
| --- |
| namespace Swiper.Utils;  internal class DescriptionGenerator  {  private readonly string[] \_adjectives = { "nice", "horrible", "great", "terribly old", "brand new" };  private readonly string[] \_other = { "picture of grandpa", "car", "photo of a forest", "duck" };  private static readonly Random random = new();  public string Generate()  {  var a = \_adjectives[random.Next(\_adjectives.Count())];  var b = \_other[random.Next(\_other.Count())];  return $"A {a} {b}";  }  } |

Utils/Picture.cs

|  |
| --- |
| namespace Swiper.Utils;  internal class Picture  {  public Uri Uri { get; init; }  public string Description { get; init; }  public Picture()  {  Uri = new Uri($"https://picsum.photos/400/400/?random&ts={ DateTime.Now.Ticks }");  var generator = new DescriptionGenerator();  Description = generator.Generate();  }  } |

MainPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="Swiper.MainPage">  <Grid Padding="0,40" x:Name="MainGrid" WidthRequest="{OnIdiom -1, *Tablet*=400, *Desktop*=400}">  <Grid.RowDefinitions>  <RowDefinition Height="400" />  <RowDefinition Height="\*" />  </Grid.RowDefinitions>  <Grid Grid.Row="1" Padding="30">  <Grid.RowDefinitions>  <RowDefinition Height="auto" />  <RowDefinition Height="auto" />  <RowDefinition Height="auto" />  <RowDefinition Height="auto" />  </Grid.RowDefinitions>  <Label Text="LIKES" />  <Label x:Name="likeLabel" Grid.Row="1" Text="0" FontSize="Large" FontAttributes="Bold" />  <Label Grid.Row="2" Text="DENIED" />  <Label x:Name="denyLabel" Grid.Row="3" Text="0" FontSize="Large" FontAttributes="Bold" />  </Grid>  </Grid>  </ContentPage> |

MainPage.xaml.cs

|  |
| --- |
| using Swiper.Controls;  namespace Swiper;  public partial class MainPage : ContentPage  {  private int \_likeCount;  private int \_denyCount;  public MainPage()  {  InitializeComponent();  AddInitialPhotos();  }  private void AddInitialPhotos()  {  for (int i = 0; i < 10; i++)  {  InsertPhoto();  }  }  private void InsertPhoto()  {  var photo = new SwiperControl();  photo.OnDeny += Handle\_OnDeny;  photo.OnLike += Handle\_OnLike;  this.MainGrid.Children.Insert(0, photo);  }  private void UpdateGui()  {  likeLabel.Text = \_likeCount.ToString();  denyLabel.Text = \_denyCount.ToString();  }  private void Handle\_OnLike(object sender, EventArgs e)  {  \_likeCount++;  InsertPhoto();  UpdateGui();  }  private void Handle\_OnDeny(object sender, EventArgs e)  {  \_denyCount++;  InsertPhoto();  UpdateGui();  }  } |