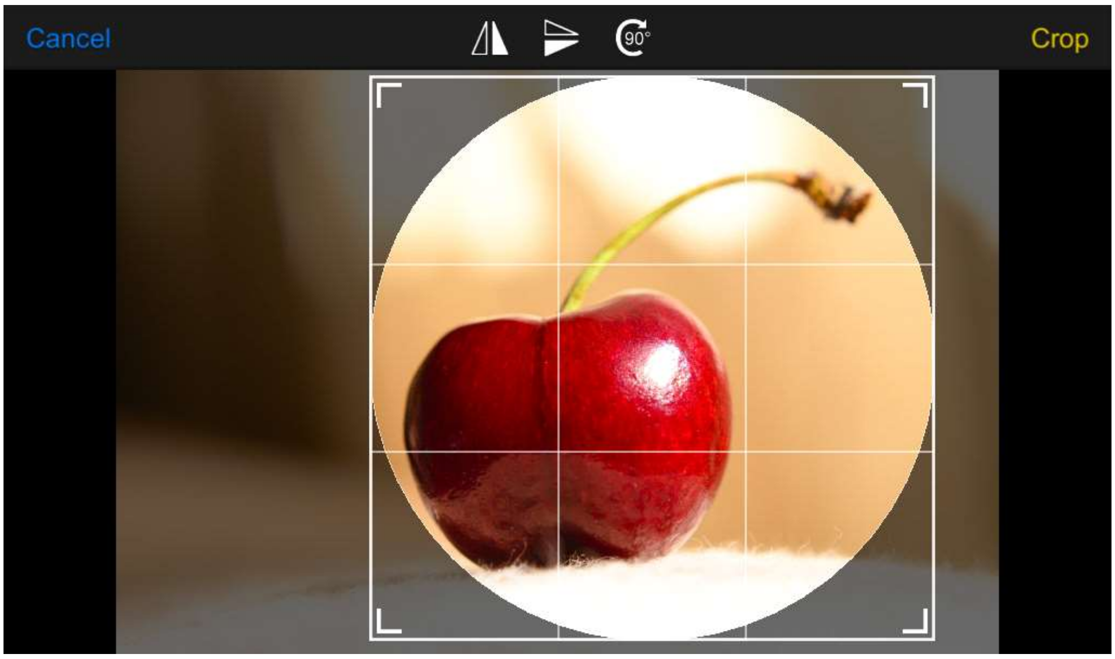
Image Cropper

E-mail: *yasirkula@gmail.com*



1. ABOUT

Image Cropper aims to be a powerful, customizable and easy-to-use image cropping solution for Unity. It is created with Unity's UI system.

2. HOW TO

ImageCropper is a singleton object that can be accessed via `ImageCropper.Instance`. To start cropping a *Texture* object, you can call the following function:

```
ImageCropper.Instance.Show( Texture image, CropResult onCrop, Settings settings
= null, ImageResizePolicy croppedImageResizePolicy = null )
```

- **image**: the Texture object to crop
- **onCrop**: callback that will be invoked after cropping is finished/cancelled. Its signature is as following: `void CropResult( bool result, Texture originalImage, Texture2D croppedImage )`. **result** stores whether the image is cropped successfully or operation is cancelled, **originalImage** stores the Texture passed as *image* parameter and **croppedImage** stores the resulting cropped image
- **croppedImageResizePolicy**: called before the cropped image is saved as *Texture2D*. Can be used to override the size of the Texture2D object. Its signature is as following: `void ImageResizePolicy( ref int width, ref int height )`, where *width* and *height* will initially be equal to the dimensions of the selection (crop area). If left *null*, texture's size will be equal to the selection's size. Can be used to e.g. always output a 256x256 Texture2D

- **settings**: can be used to adjust the parameters of the image cropper. Available parameters are:

  i. **bool autoZoomEnabled** (*default=true*): if enabled, image cropper will zoom in to the selection (crop area) if it is too small, and zoom out if selection is too large

  ii. **bool pixelPerfectSelection** (*default=false*): if enabled, selection's position and size values will be rounded to the nearest integers. As a *RenderTexture* with a render camera is used to generate the cropped image (instead of reading the pixels of the source image), a pixel perfect selection doesn't really have a big impact on the output

  iii. **bool ovalSelection** (*default=false*): if enabled, an oval mask will be used to crop the image in oval/circular shape. Otherwise, image will be cropped in rectangular/square shape

  iv. **bool markTextureNonReadable** (*default=true*): marks the cropped texture as non-readable for better memory usage. If you plan to modify the texture later (e.g. *GetPixels/SetPixels*), set its value to *false*

  v. **Color imageBackground** (*default=black*): determines the background color of the cropped image. Background color will be visible if source image has transparency or if *ovalSelection* is enabled. For a completely transparent background, set its value to *Color.clear* (which has 0 alpha). Note that if imageBackground is opaque (alpha=1), cropped texture will be in *RGB24* format instead of *RGBA32* format. As RGB24 uses less memory, try not to use a transparent background color unless it is needed

  vi. **Button visibleButtons** (*default=FlipHorizontal|FlipVertical|Rotate90Degrees*): determines which image orientation buttons will be visible in the user interface. By default, all buttons are visible

  vii. **Visibility guidelinesVisibility** (*default=AlwaysVisible*): determines the visibility of the selection guidelines. Accepted values are: *Hidden*, *OnDrag* (only visible while the selection is being dragged/resized) and *AlwaysVisible*

  viii. **Orientation initialOrientation** (*defaul=Normal*): initial orientation (flipped/rotated state) of the image. Please see EXIF orientations before changing its value: http://sylvana.net/jpegcrop/exif_orientation.html

  ix. **Vector2 selectionMinSize** (*default=0,0*): minimum size of the selection (crop area). If untouched, it will be equal to 1/10th of the source image's size

  x. **Vector2 selectionMaxSize** (*default=0,0*): maximum size of the selection (crop area). If untouched, there will be no limit

  xi. **float selectionMinAspectRatio** (*default=0*): minimum aspect ratio of the selection (crop area). If untouched, there will be no limit

  xii. **float selectionMaxAspectRatio** (*default=0*): maximum aspect ratio of the selection (crop area). If untouched, there will be no limit. For a circular/square selection, you can set both *selectionMinAspectRatio* and *selectionMaxAspectRatio* to 1

xiii. **float selectionInitialPaddingLeft** (*default=0.1*): initial padding-left of the selection in %

xiv. **float selectionInitialPaddingTop** (*default=0.1*): initial padding-top of the selection in %

xv. **float selectionInitialPaddingRight** (*default=0.1*): initial padding-right of the selection in %

xvi. **float selectionInitialPaddingBottom** (*default=0.1*): initial padding-bottom of the selection in %

NOTE: before calling the *Show* function, you may want to check the value of `ImageCropper.Instance.IsOpen` to make sure that the image cropper is not already visible