## Part 1

For this part of the project, it is required us to detect dices and press a specific key based on the number of circles on the dice. In order to count the circles, first we need to crop out the dice squares first to get a better results. In the first game, the dices are standing still, with one side is being shown. In the second game the dices are rotating with respect to a specific axis. We come up with a robust algorithm that works for both game 1 and 2.

Our method is as follows:

1. Take the screenshot first, if the game is the second one, we take 3. Then convert it to grayscale image, and threshold pixels greater than 200(to detect the dice faces);

```
img = takeScreenShot(filename="part1.png", toSave=False)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
mask = np.uint8((gray >= 200)*255.0)
```

2. After thresholding the dices, we apply connected component labelling. After getting the labels, second, third and fourth labels are matched with the dice faces. The reason we took 2nd, 3rd and 4th is because the first label is the background. By matching and cropping out the dices, we can proceed to counting circles.

```
 _, labels = cv2.connectedComponents(mask)

 dices = []
 for i in range(1,4):
     dice_mask = np.uint8((labels == i) * 255.)
     dice_mask = cv2.GaussianBlur(dice_mask, (35,35), 1)

     corners =
np.array(cv2.goodFeaturesToTrack(dice_mask,4,0.001,300).reshape(-1, 2))

     tl_y = int(np.min(corners[:,0]))
     tl_x = int(np.min(corners[:,1]))
     br_y = int(np.max(corners[:,0]))
     br_x = int(np.max(corners[:,1]))

     dice = np.uint8(gray[tl_x:br_x, tl_y:br_y])

     dices.append(dice)
```

3. After cropping out the dice faces, we start counting the circles in the dice faces. Finally, we press to a specific key based on which dice face has the greatest number of circles.

```
 circle_count = []
 for i, dice in enumerate(dices):
     circles = cv2.HoughCircles(dice, cv2.HOUGH_GRADIENT, 1, dice.shape[0] / 4,
param1=200, param2=10, minRadius=20, maxRadius=50)
     if circles is not None:
```

```
            circle_count.append(circles.shape[1])
        else:
            circle_count.append(0)
        ind = np.argmax(circle_count)
```

The difference between first game and second game, we needed to make a perspective transform in order to get dice faces like in part 1. Since there are multiple faces seen in the game, we have selected the face with highest area. In order to process them in order like it appears in the game, we firstly sorted based on its pixel size and select second third and fourth, then sorted based on the minimum x coordinate value. This way we can guarantee that we get dice face with largest area in order as it appears in screen. After that, we calculate the corner coordinates, and apply perspective transform to get a dice face like in game 1.