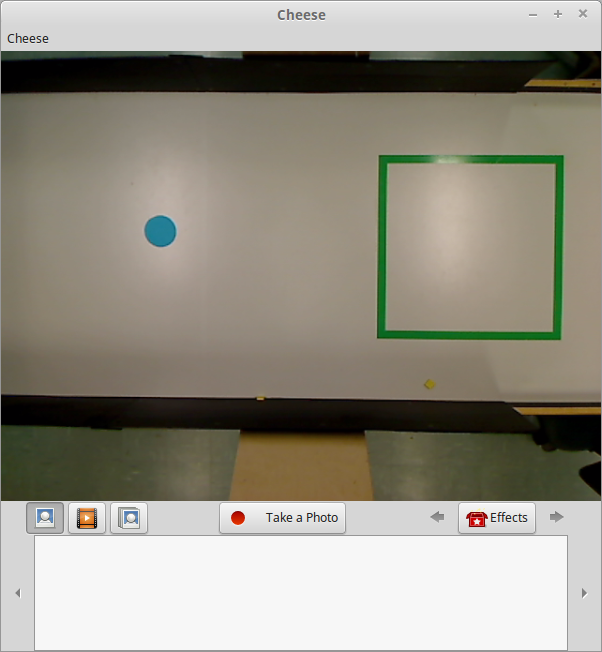
# Acquisition des images à l'aide de la caméra World

Afin de se créer une banque d'images pour créer des tests automatisés, nous devions faire fonctionner la caméra world sur un ordinateur. Le premier test effectué a été de prendre des photos à l'aide d'un logiciel pour webcam. Nous avons utilisé le logiciel Linux **Cheese**. Voici le résultat:



La deuxième étape de cette expérimentation a été de prendre une image à l'aide d'OpenCV. Pour accomplir cela, nous avons utilisé la fonction Python suivante:

camera = cv2.VideoCapture(self.camera\_port)

value, image\_capture = camera.read()

Nous avons aussi réussi à change la résolution à l'aide des paramètres suivant avant la prise de photo:

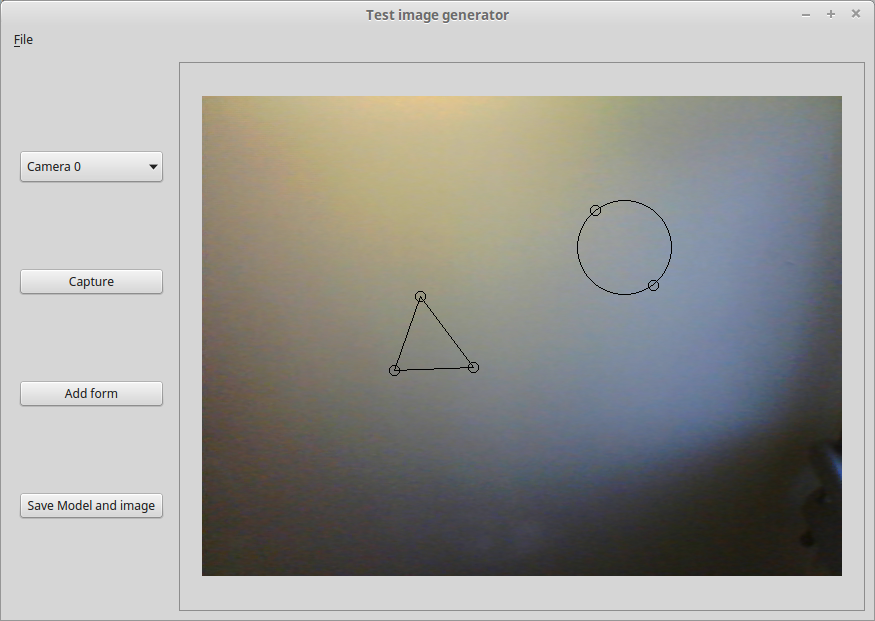
camera.set(cv2.cv.CV\_CAP\_PROP\_FRAME\_WIDTH, 1600)

camera.set(cv2.cv.CV\_CAP\_PROP\_FRAME\_HEIGHT, 1200)

# Création d'une banque d'image de tests

Pour la création d'une banque de données, nous avons programmé un petit logiciel simple qui permet de prendre des photos avec la webcam et ensuite d'identifier les formes qui se trouvent sur l'image ainsi que leur couleur.

Voici une capture d'écran de ce logiciel:

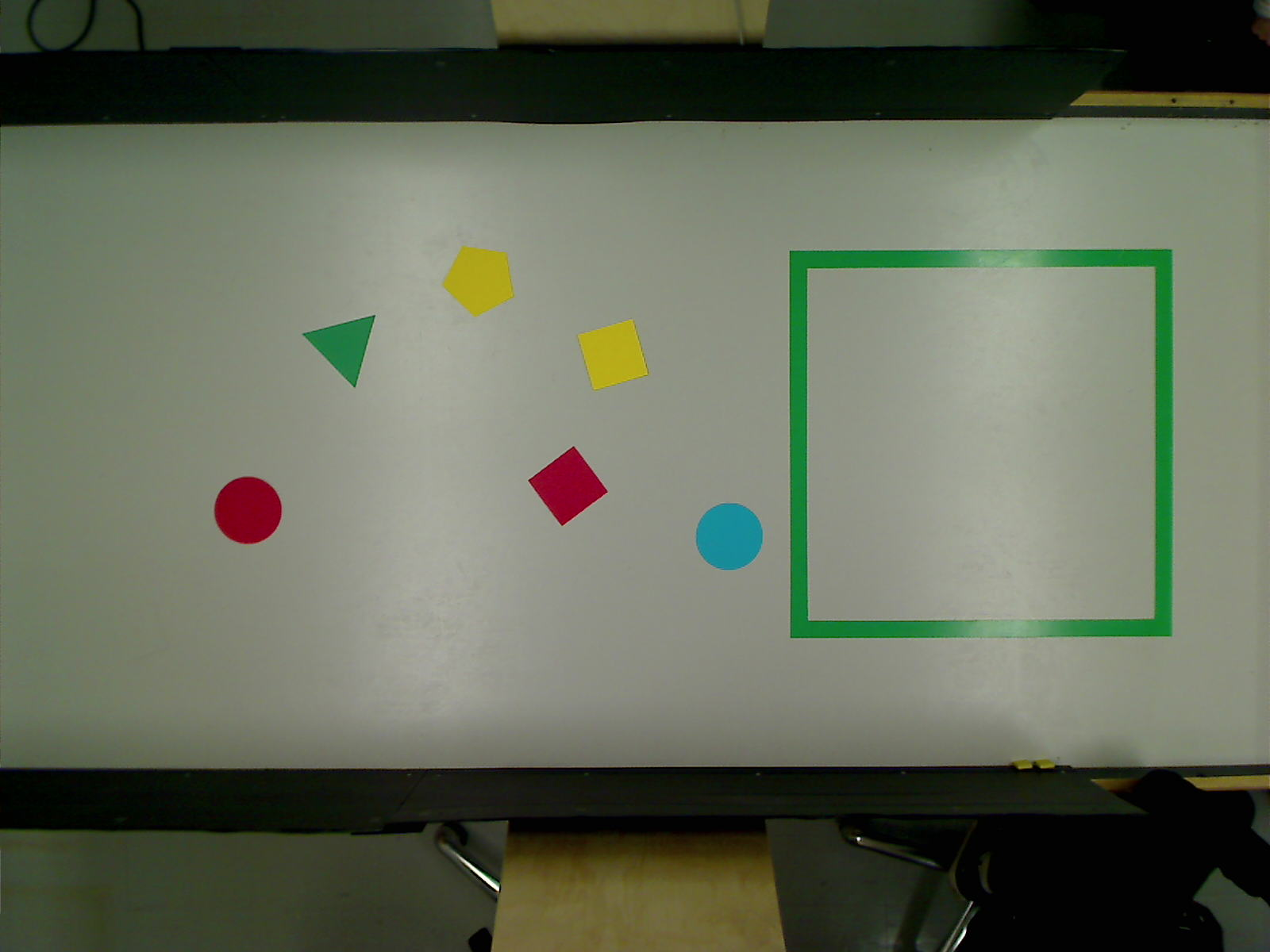


Après avoir pris la photo et identifié les formes, on appuie sur “Save Model and image” et le logiciel enregistre une image “timestamp.png” ainsi qu'un fichier “timestamp.json” qui contient la description des formes et leur position. Le fichier contient aussi la position et l'orientation du robot mais l'enregistrement de cette information n'a toujours pas été implémentée puisque nous n'avons pas définitivement décidé le mode de localisation du robot encore.

Les fichiers json et png sont ensuite envoyés sur notre repositoire Git où ils seront utilisés lorsque nous élaborerons nos tests automatisés de vision.

# Détection de formes sur OpenCV

Après avoir capturé des images avec la caméra world, nous avons expérimenté avec la détection de formes sur OpenCV. Voici nos résultats:

 Image initiale Image avec un masque sur le rouge

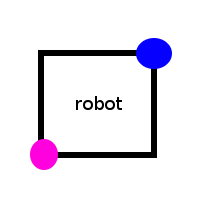
Et voici le résultat avec la détection des formes (entouré de jaune signifie que le logiciel a détecté un cercle et entouré de rouge signifie que le logiciel a détecté un carré)

Ceci n'est qu'un premier jet vers la détection fiable d'objets par OpenCV. En effet, cette détection a été effectuée en calculant le nombre de côtés de chaque forme ce qui ne fournissait pas des résultats très fiables, par exemple, dépendemment du blur utilisé, les formes avaient un nombre très différent de côtés. Nous allons tester d'autres techniques de détection telles que l'angle entre les arrêtes ou une combinaison de techniques.

# Recherches sur les moyens de détection du robot

Des recherches brèves ont aussi été effectuées pour trouver un moyen facile et fiable de déterminer la position et l'orientation du robot en tout temps.

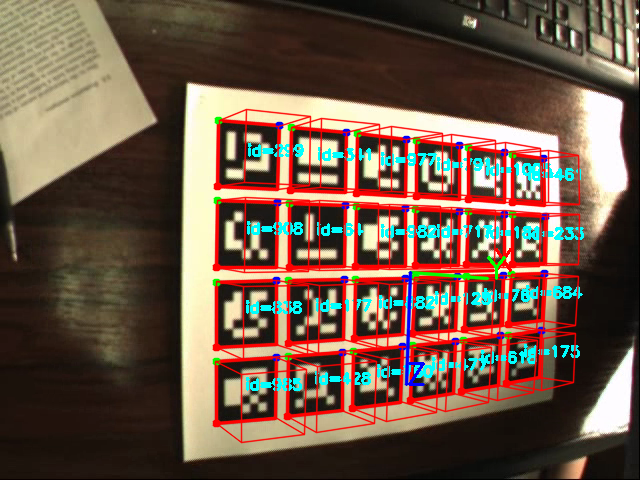
La première option envisagée est de fixer des marqueurs de couleurs différentes que celles des îles aux coins du robot comme ceci:



Cette technique requiert toutefois de calculer l'orientation du robot à l'aide de la position des balles. De plus, les calculs de positions et d'orientation requiert une position exacte des deux marqueurs en tout temps ce qui rend cette solution fragile.

Nous avons ensuite trouvé une librairie, ArUco, qui permet de détecter facilement des objets à l'aide de tableaux ressemblant à des codes QR.

<http://www.uco.es/investiga/grupos/ava/node/26>



À l'aide de cette librairie, on obtient une précision plus grande dû au plus grand nombre de marqueurs et cette méthode fournit aussi une redondance et un calcul de position et d'orientation automatique. Nous considérons donc grandement cette librairie pour effectuer la détection de notre robot.