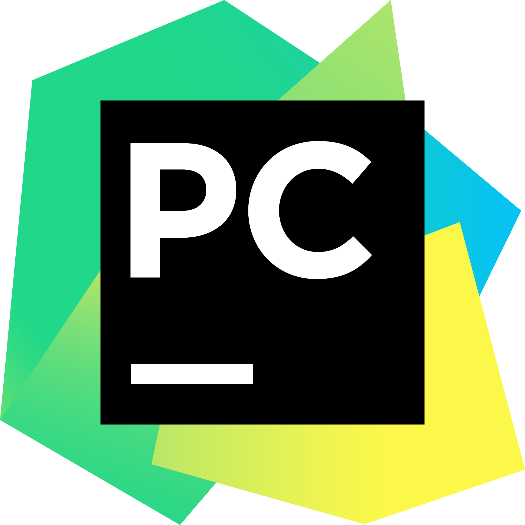
**Process Mise en place de l’environnement de développement du projet :**

BUG Touch

Objectif : Préparer l’environnement de développement

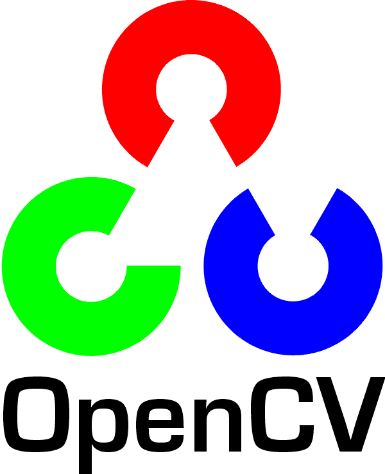
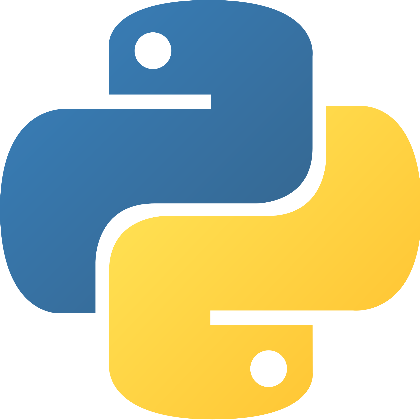


Sommaire des étapes :

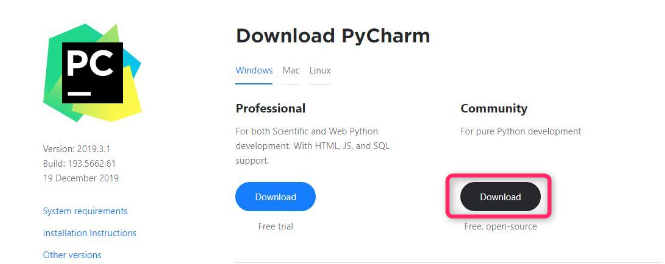
1. Installation de l’environnement PyCharm (python)

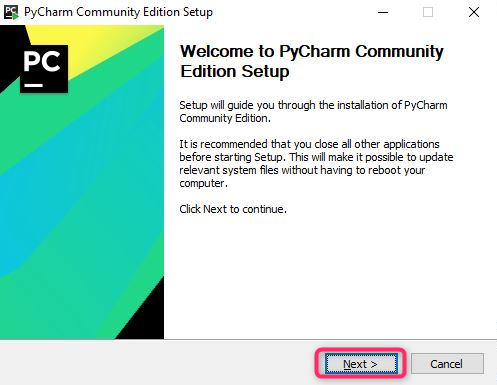
2. Récupération du Code et Installation des bibliothèques

- OpenCV - Mediapipe - PyGame

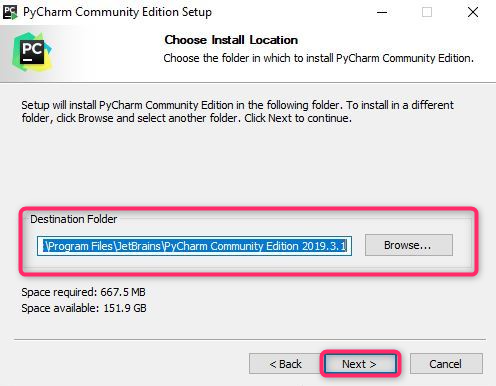
3. Test de l’environnement de développement

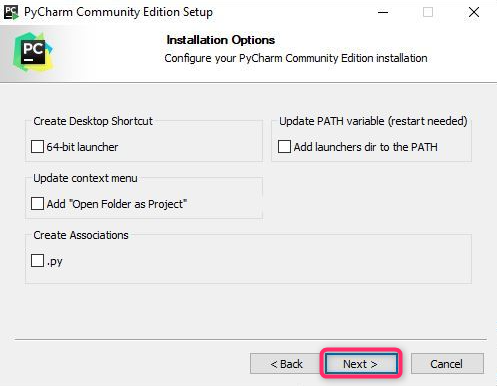
1. Installation de l’environnement PyCharm

Tout d’abord télécharger PyCharm sur [jetbains.com](https://www.jetbrains.com/pycharm/download/#section=windows)

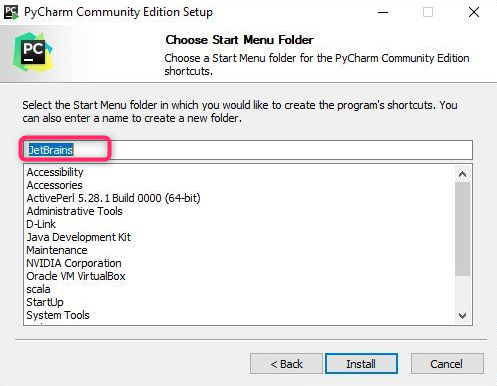
Télécharger la version Community de PyCharm et suivre les instructions d’installations.

Choix de l’emplacement d’installation de PyCharm.

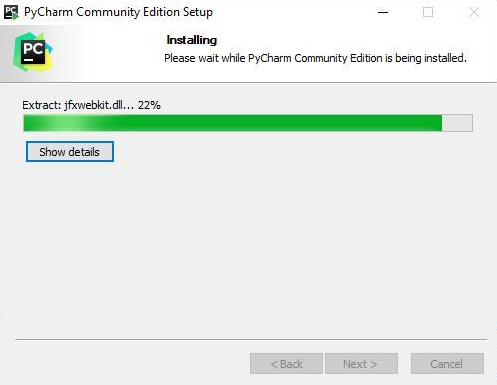


Options d’installation, faite Next directement.

Ici on sélectionnera ‘’JetBrains’’ comme dossier du menu démarrer



Installation en cours, il nous reste plus qu’à attendre.



Installation terminée !

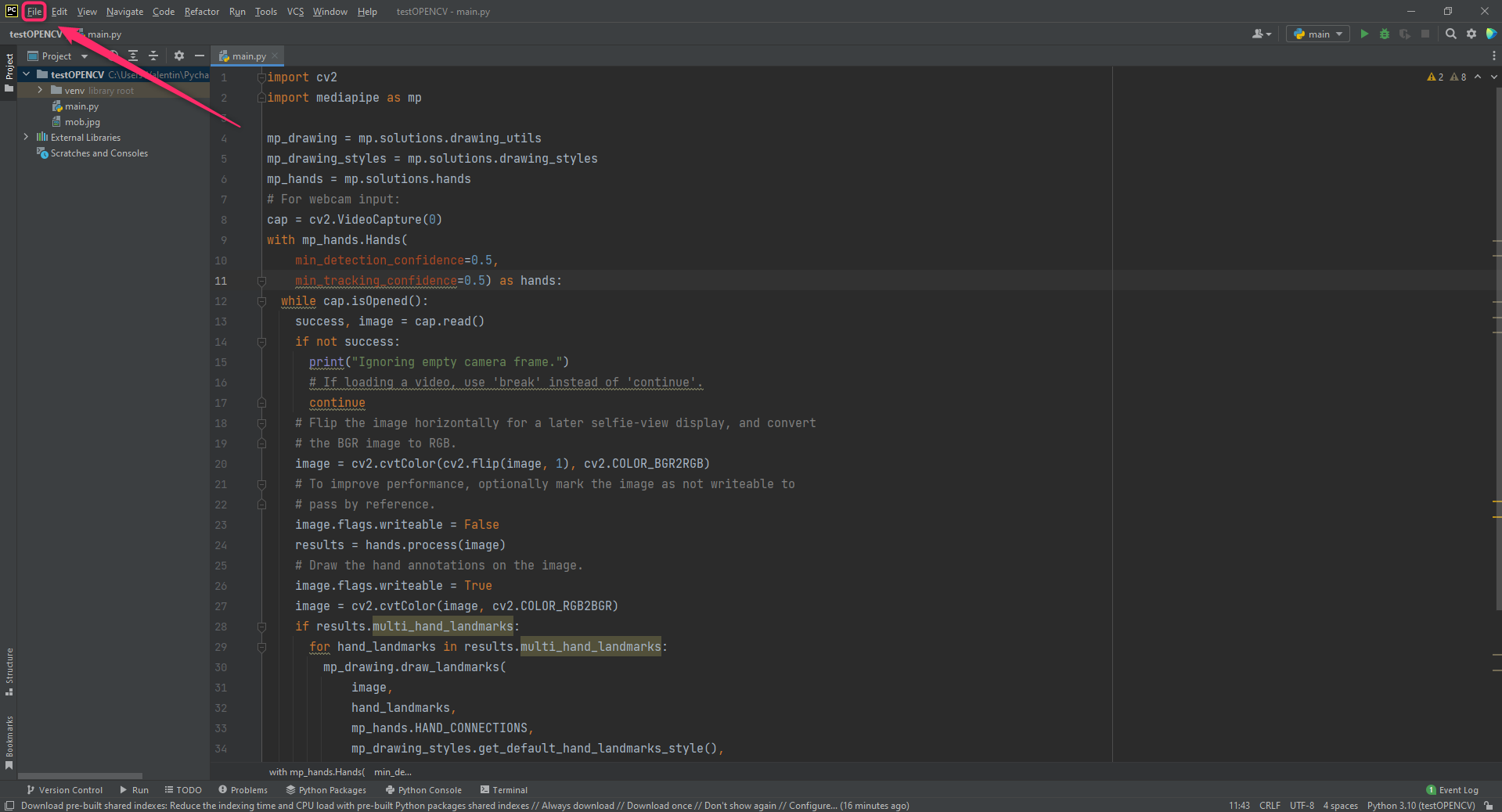
Maintenant que l’IDE PyCharm est installer il faut maintenant récupérer le code sur le GIT du projet et installer les bibliothèques nécessaires au bon fonctionnement du programme.

2. Récupération du Code et Installation des bibliothèques

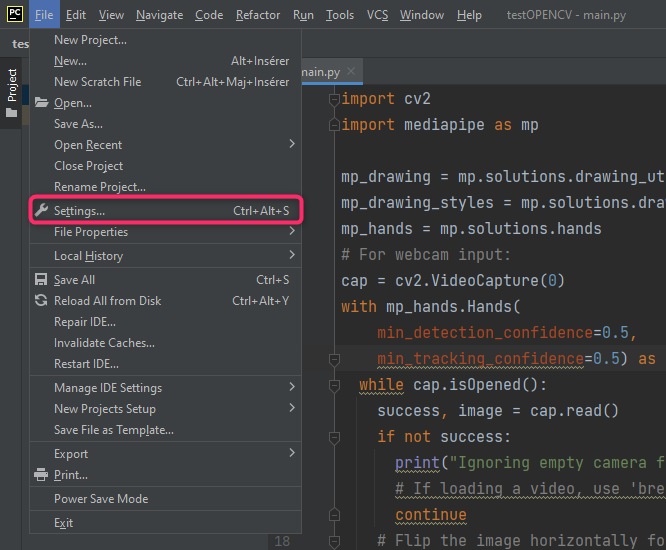
Voici un lien qui mène à une excellente vidéo montrant comment récupérer le code sur GIT et directement l’inclure dans PyCharm

Lien de la vidéo : [Récupérer le code d’un projet GIT PyCharm](https://www.youtube.com/watch?v=mX4h3UTCV7g)

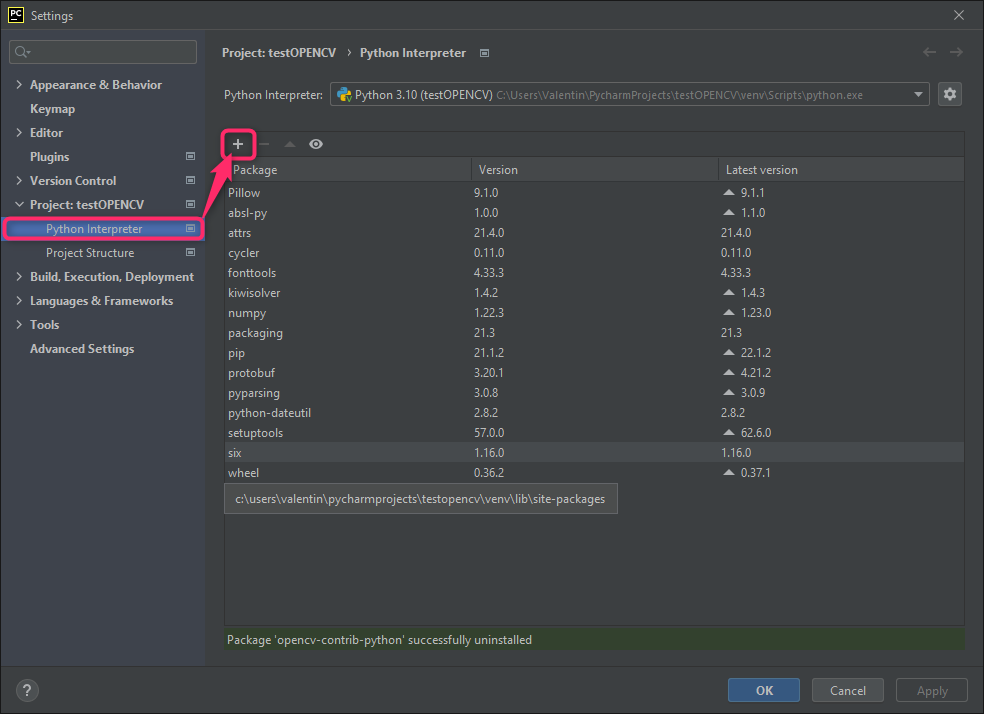
Maintenant le code récupérer et ouvert sur PyCharm, il faut installer la bibliothèque OpenCV, Mediapipe et PyGame afin de pouvoir lancer le programme et donc de pouvoir commencer à travailler !

Aller dans l’onglet ‘’File’’

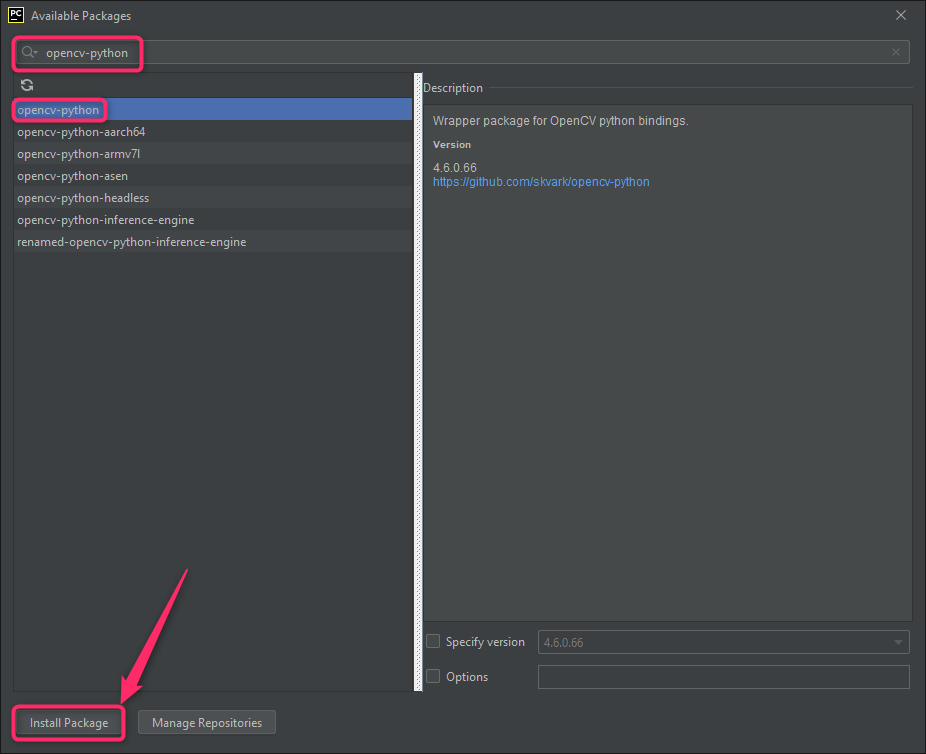
Une fois dans l’onglet ‘’File’’ cliquer sur ‘’Settings’’

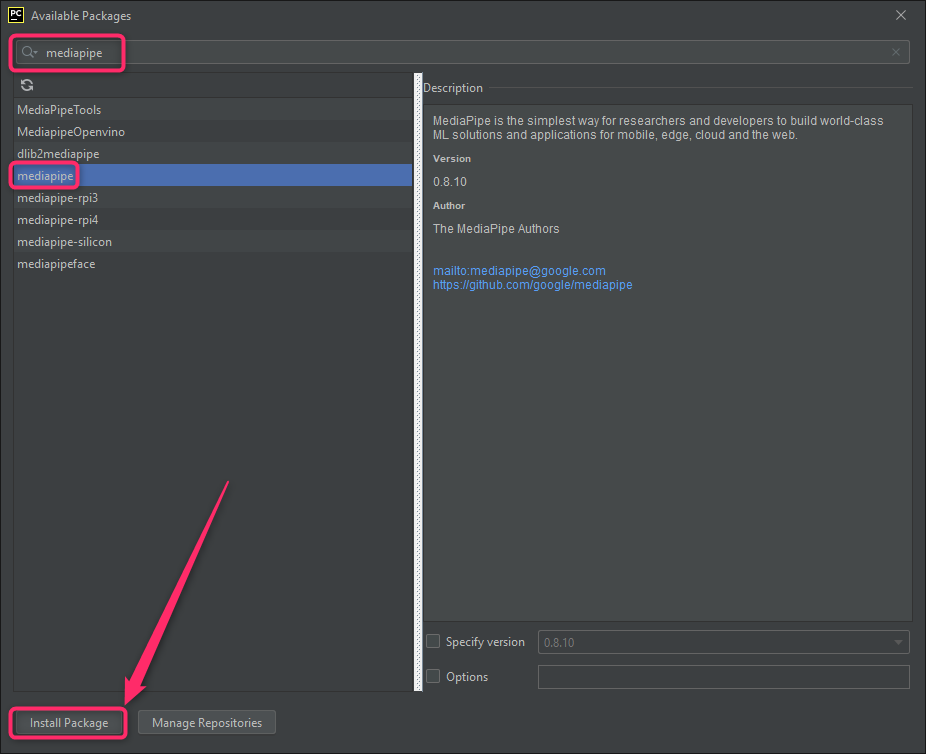


Une fois dans ‘’Settings’’ sélectionner ‘’Python interpreter’’ puis sur le ‘’+’’ pour ajouter les bibliothèques

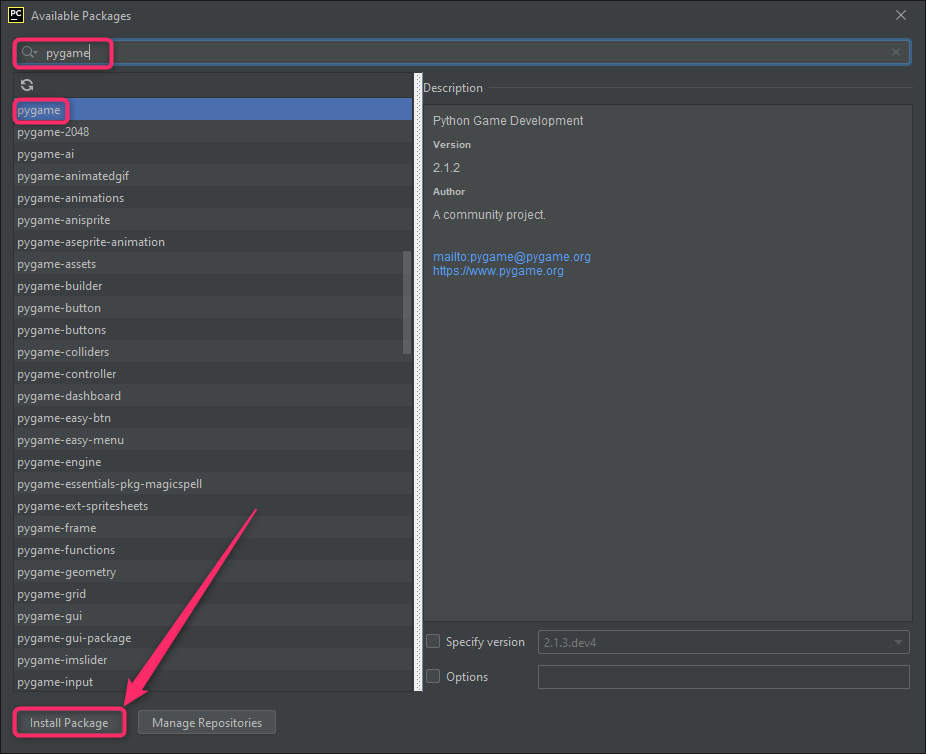


Dans le nouvel onglet qui s’ouvre, taper ‘’opencv-python’’ et sélectionner ‘’opencv-python’’ et appuyer sur ‘’install package’’



Faire de même avec ‘’mediapipe’’

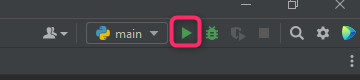
Et faire de même avec ‘’PyGame’’

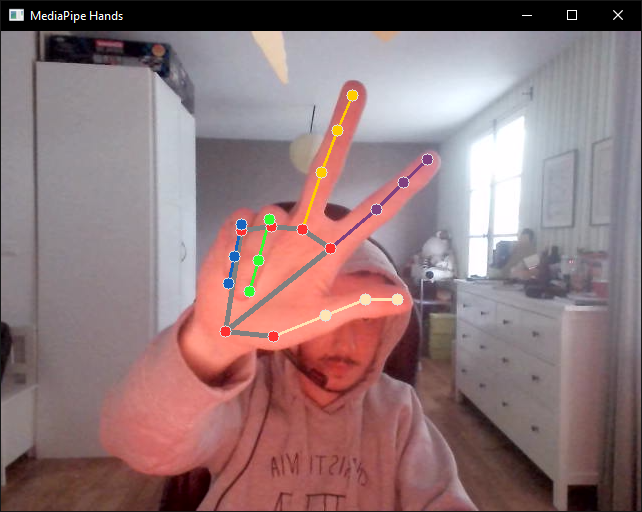


3. Test de l’environnement de développement

Maintenant que ces bibliothèques sont installées nous pouvons les tester histoire de voir si elles sont vraiment utilisables et bien installer.

Lien d’un [Programme de Test pour OpenCV et Mediapipe](#_CODE_DE_TEST_1), celui-ci devrait ouvrir une nouvelle fenêtre via la caméra de l’ordinateur. Si celle-ci s’ouvre bien alors vous devriez avoir la même chose que ci-dessous. Si c’est le cas alors tout est bon !



Pour lancer le programme, c’est la flèche verte en haut à droite de la fenêtre.

Maintenant il nous faut voir si **PyGame** fonctionne bien. Pour le savoir il faudra faire comme pour **OpenCV** et **Mediapipe**. Tenter de lancer le [Programme ci-dessous](#_CODE_DE_TEST_2)**.**

**Une fois le programme lancer, vous devriez avoir l’affichage vvvv suivant vvvv**

# 

Si tout à fonctionner jusqu’ici alors tout est prêt pour travailler sur notre projet **BUG** **Touch**, félicitations !

# **CODE DE TEST OPENCV MEDIAPIPE**

import cv2  
import mediapipe as mp  
  
mp\_drawing = mp.solutions.drawing\_utils  
mp\_drawing\_styles = mp.solutions.drawing\_styles  
mp\_hands = mp.solutions.hands  
# For webcam input:  
cap = cv2.VideoCapture(0)  
with mp\_hands.Hands(  
 min\_detection\_confidence=0.5,  
 min\_tracking\_confidence=0.5) as hands:  
 while cap.isOpened():  
 success, image = cap.read()  
 if not success:  
 print("Ignoring empty camera frame.")  
 # If loading a video, use 'break' instead of 'continue'.  
 continue  
 # Flip the image horizontally for a later selfie-view display, and convert  
 # the BGR image to RGB.  
 image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR\_BGR2RGB)  
 # To improve performance, optionally mark the image as not writeable to  
 # pass by reference.  
 image.flags.writeable = False  
 results = hands.process(image)  
 # Draw the hand annotations on the image.  
 image.flags.writeable = True  
 image = cv2.cvtColor(image, cv2.COLOR\_RGB2BGR)  
 if results.multi\_hand\_landmarks:  
 for hand\_landmarks in results.multi\_hand\_landmarks:  
 mp\_drawing.draw\_landmarks(  
 image,  
 hand\_landmarks,  
 mp\_hands.HAND\_CONNECTIONS,  
 mp\_drawing\_styles.get\_default\_hand\_landmarks\_style(),  
 mp\_drawing\_styles.get\_default\_hand\_connections\_style())  
 cv2.imshow('MediaPipe Hands', image)  
 if cv2.waitKey(5) & 0xFF == 27:  
 break  
cap.release()

# **CODE DE TEST PYGAME (IHM)**

A récupérer sur le GIT (surtout les .**png** pour les textures)

import pygame, random, sys  
from pygame.locals import \*  
  
pygame.init()  
  
window = pygame.display.set\_mode((1280, 720), HWSURFACE | DOUBLEBUF | RESIZABLE)  
cursor = pygame.image.load(r'cursor.png')  
cursor = pygame.transform.scale(cursor,(50,50))  
background = pygame.image.load(r'fond.png')  
pygame.display.set\_caption("Insect Touch")  
  
window.blit(pygame.transform.scale(background, (1280, 720)), (0, 0))  
  
pygame.mouse.set\_visible(False)  
  
class Button():  
 def \_\_init\_\_(self, image, pos, text\_input, font, base\_color, hovering\_color):  
 self.image = image  
 self.x\_pos = pos[0]  
 self.y\_pos = pos[1]  
 self.font = font  
 self.base\_color, self.hovering\_color = base\_color, hovering\_color  
 self.text\_input = text\_input  
 self.text = self.font.render(self.text\_input, True, self.base\_color)  
 if self.image is None:  
 self.image = self.text  
 self.rect = self.image.get\_rect(center=(self.x\_pos, self.y\_pos))  
 self.text\_rect = self.text.get\_rect(center=(self.x\_pos, self.y\_pos))  
  
 def update(self, screen):  
 if self.image is not None:  
 screen.blit(self.image, self.rect)  
 screen.blit(self.text, self.text\_rect)  
  
 def checkForInput(self, position):  
 if position[0] in range(self.rect.left, self.rect.right) and position[1] in range(self.rect.top,  
 self.rect.bottom):  
 return True  
 return False  
  
 def changeColor(self, position):  
 if position[0] in range(self.rect.left, self.rect.right) and position[1] in range(self.rect.top,  
 self.rect.bottom):  
 self.text = self.font.render(self.text\_input, True, self.hovering\_color)  
 else:  
 self.text = self.font.render(self.text\_input, True, self.base\_color)  
  
def get\_font(size): # Returns Press-Start-2P in the desired size  
 return pygame.font.Font(r"font.ttf", size)  
  
def show\_score(x,y,score\_value):  
  
 score = get\_font(60).render("Score : "+str(score\_value), True,"#b68f40")  
 window.blit(score,(x,y))  
  
def main\_menu(window): # Menu du Jeu  
  
 pygame.display.set\_caption("Menu")  
  
 while True:  
  
 MENU\_TEXT = get\_font(100).render("BUG Touch", True, "#b68f40")  
 MENU\_RECT = MENU\_TEXT.get\_rect(center=((window.get\_width()/2), window.get\_height()/9))  
 MENU\_BACKGROUND\_IMAGE = pygame.image.load(r'Play Rect.png')  
 MENU\_BACKGROUND = pygame.transform.scale(MENU\_BACKGROUND\_IMAGE,(950,125))  
  
 BOUTTON\_JOUER = Button(image=pygame.image.load(r'Play Rect.png'),pos=((window.get\_width()/2),window.get\_height()/2-125),  
 text\_input='JOUER', font=get\_font(50),base\_color="#d7fcd4", hovering\_color="White")  
 BOUTTON\_REGLES = Button(image=pygame.image.load(r'Play Rect.png'), pos=((window.get\_width()/2), window.get\_height()/2+25),  
 text\_input="REGLES", font=get\_font(50), base\_color="#d7fcd4", hovering\_color="White")  
 BOUTTON\_QUITTER = Button(image=pygame.image.load(r'Play Rect.png'), pos=((window.get\_width()/2), window.get\_height()/2+175),  
 text\_input="QUITTER", font=get\_font(50), base\_color="#d7fcd4", hovering\_color="White")  
  
 window.blit(MENU\_BACKGROUND,((window.get\_width()/2)/4, (window.get\_height()/2)/30))  
 window.blit(MENU\_TEXT,MENU\_RECT)  
  
 for button in [BOUTTON\_JOUER, BOUTTON\_REGLES, BOUTTON\_QUITTER]:  
 button.changeColor(pygame.mouse.get\_pos())  
 button.update(window)  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
  
 elif event.type == VIDEORESIZE:  
 if window.get\_width()<1152:  
 window = pygame.display.set\_mode((1152,window.get\_height()), pygame.RESIZABLE)  
 pygame.display.update()  
 elif window.get\_height()<648:  
 window = pygame.display.set\_mode((window.get\_width(), 648), pygame.RESIZABLE)  
 pygame.display.update()  
 else:  
 window = pygame.display.set\_mode(event.dict['size'], HWSURFACE | DOUBLEBUF | RESIZABLE)  
 window.blit(pygame.transform.scale(background, event.dict['size']), (0, 0))  
 pygame.display.flip()  
  
 if event.type == pygame.MOUSEBUTTONDOWN:  
 if BOUTTON\_JOUER.checkForInput(pygame.mouse.get\_pos()):  
 jouer(window)  
 if BOUTTON\_REGLES.checkForInput(pygame.mouse.get\_pos()):  
 regles(window)  
 if BOUTTON\_QUITTER.checkForInput(pygame.mouse.get\_pos()):  
 pygame.quit()  
 sys.exit()  
  
 window.blit(cursor, pygame.mouse.get\_pos())  
  
 pygame.display.update()  
 window.blit(pygame.transform.scale(background, (window.get\_width(), window.get\_height())), (0, 0))  
  
  
def jouer(window):  
  
 SCORE\_BACKGROUND\_IMAGE = pygame.image.load(r'Play Rect.png')  
 SCORE\_BACKGROUND = pygame.transform.scale(SCORE\_BACKGROUND\_IMAGE, (700,85))  
  
 score\_value = 0  
 rect = pygame.Rect(60, 60, 60, 60);  
  
 while True:  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 pygame.quit()  
 sys.exit()  
  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_ESCAPE:  
 main\_menu(window)  
  
 elif event.type == VIDEORESIZE:  
 if window.get\_width()<1152:  
 window = pygame.display.set\_mode((1152,window.get\_height()), pygame.RESIZABLE)  
 pygame.display.update()  
 elif window.get\_height()<648:  
 window = pygame.display.set\_mode((window.get\_width(), 648), pygame.RESIZABLE)  
 pygame.display.update()  
 else:  
 window = pygame.display.set\_mode(event.dict['size'], HWSURFACE | DOUBLEBUF | RESIZABLE)  
 window.blit(pygame.transform.scale(background, event.dict['size']), (0, 0))  
 pygame.display.update()  
  
 if event.type == pygame.MOUSEMOTION:  
 x, y = event.pos  
 if rect.collidepoint(event.pos):  
 x = random.randrange(window.get\_width())  
 y = random.randrange(window.get\_height())  
 rect.center = (x, y);  
 score\_value += 1  
 elif rect.x > window.get\_width() or rect.y > window.get\_height():  
 x = random.randrange(window.get\_width())  
 y = random.randrange(window.get\_height())  
 rect.center = (x, y);  
  
  
 window.blit(pygame.transform.scale(background, (window.get\_width(), window.get\_height())), (0, 0))  
  
 pygame.draw.circle(window, (255, 255, 0), rect.center, 50, 50)  
 window.blit(SCORE\_BACKGROUND, ((window.get\_width() / 120), window.get\_height() / 50))  
 show\_score(30, 30,score\_value)  
  
 window.blit(cursor, pygame.mouse.get\_pos())  
 pygame.display.update()  
  
def regles(window):  
  
 while True:  
  
 fond\_RETOUR = pygame.image.load(r'Play Rect.png')  
  
 BOUTTON\_RETOUR = Button(image=pygame.transform.scale(fond\_RETOUR, (300,100)),  
 pos=((window.get\_width() / 7.75), window.get\_height() / 1.1 ),  
 text\_input='RETOUR', font=get\_font(40), base\_color="#d7fcd4", hovering\_color="White")  
  
 BOUTTON\_RETOUR.changeColor(pygame.mouse.get\_pos())  
 BOUTTON\_RETOUR.update(window)  
  
 for event in pygame.event.get():  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_ESCAPE:  
 main\_menu(window)  
  
 if event.type == pygame.MOUSEBUTTONDOWN:  
 if BOUTTON\_RETOUR.checkForInput(pygame.mouse.get\_pos()):  
 main\_menu(window)  
  
 window.blit(cursor, pygame.mouse.get\_pos())  
  
 pygame.display.update()  
 window.blit(pygame.transform.scale(background, (window.get\_width(), window.get\_height())), (0, 0))  
  
main\_menu(window)