６班：最終発表

19C1041　熊谷　和真
19C1050　酒谷　冬芽
19C1068　髙橋　祐樹
19C1107　ホンスワン　タナポル
19C1138　中村　亮

目次

テーマ

- カメラで顔を認識しCrane_x7を使って人を起こす。

- 起きない場合は嫌がらせをする

どうやって起こすか

- 起こす方法
- タンバリンを叩き大きな音を出す

- 起きないときの嫌がらせ
- 重要書類をぐしゃぐしゃにする

- 人を模した人形を叩く

予定表変更前

担当者	日付	11/23	11/30	12/7	12/14	12/21
タナポル	センサの実験					
中村亮	たたくパターン１					
高橋祐樹	たたくパターン2					
熊谷和真	たたくパターン3					
酒谷冬芽	組み合わせ＋動作確認＋パッケージ化					発表

予定変更後

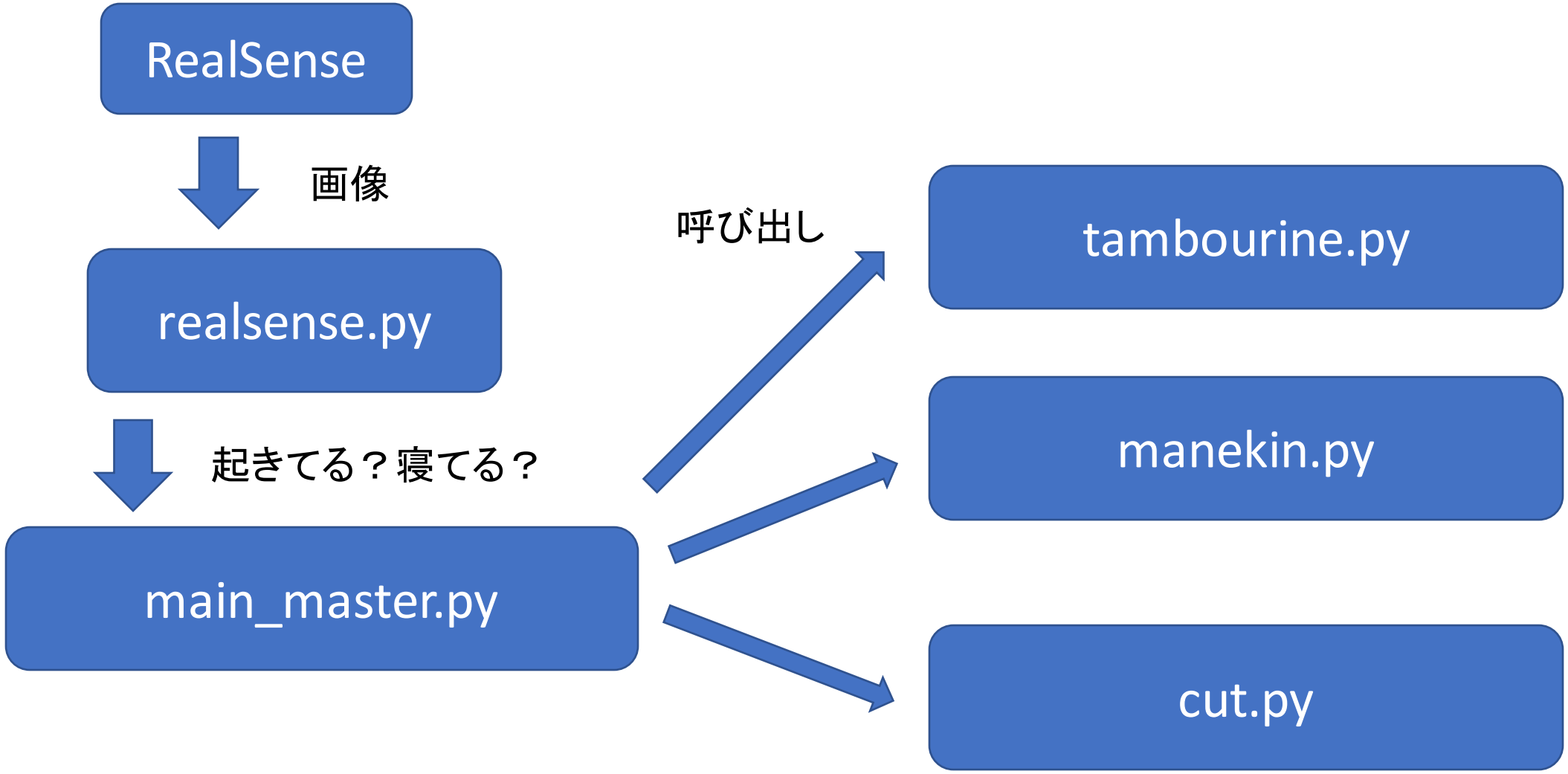
テーマ変更のため主に12月に作業を行った

担当者	日付	11/23	11/30	12/7	12/14	12/21
タナボル	センサの実験					
中村亮	音を鳴らす					
高橋祐樹	人形を叩く					
熊谷和真	重要書類をぐちゃぐちゃ					
酒谷冬芽	組み合わせ＋動作確認＋パッケージ化					発表

使用した道具

- Crane_x7
- RealSense
- 割り箸
- タンバリン
- マネキンヘッド
- A4用紙

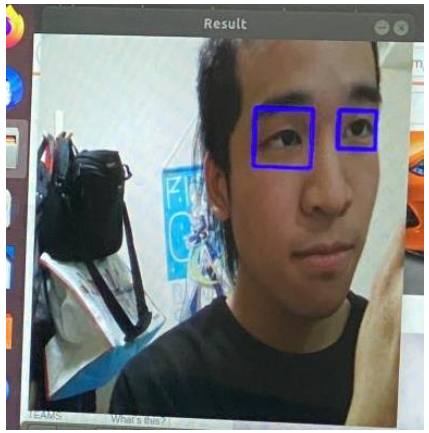
システムの構成


RealSense
画像
realsense.py
起きてる？寝てる？
main_master.py
呼び出し
tambourine.py
manekin.py
cut.py

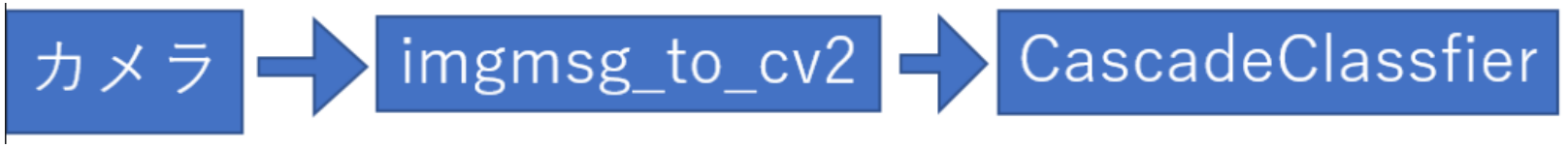
ソースコード解説（タナポル）

- RealSenseによって検出のソースコード

イケメンの目認識



eyeCascade = cv2.CascadeClassifier('ファイル場所')
eyes = eyesCascade.detectMultiScale(画像,検出させる広さ)
eyes = [検出した場所等],[]...



ソースコード解説（中村）

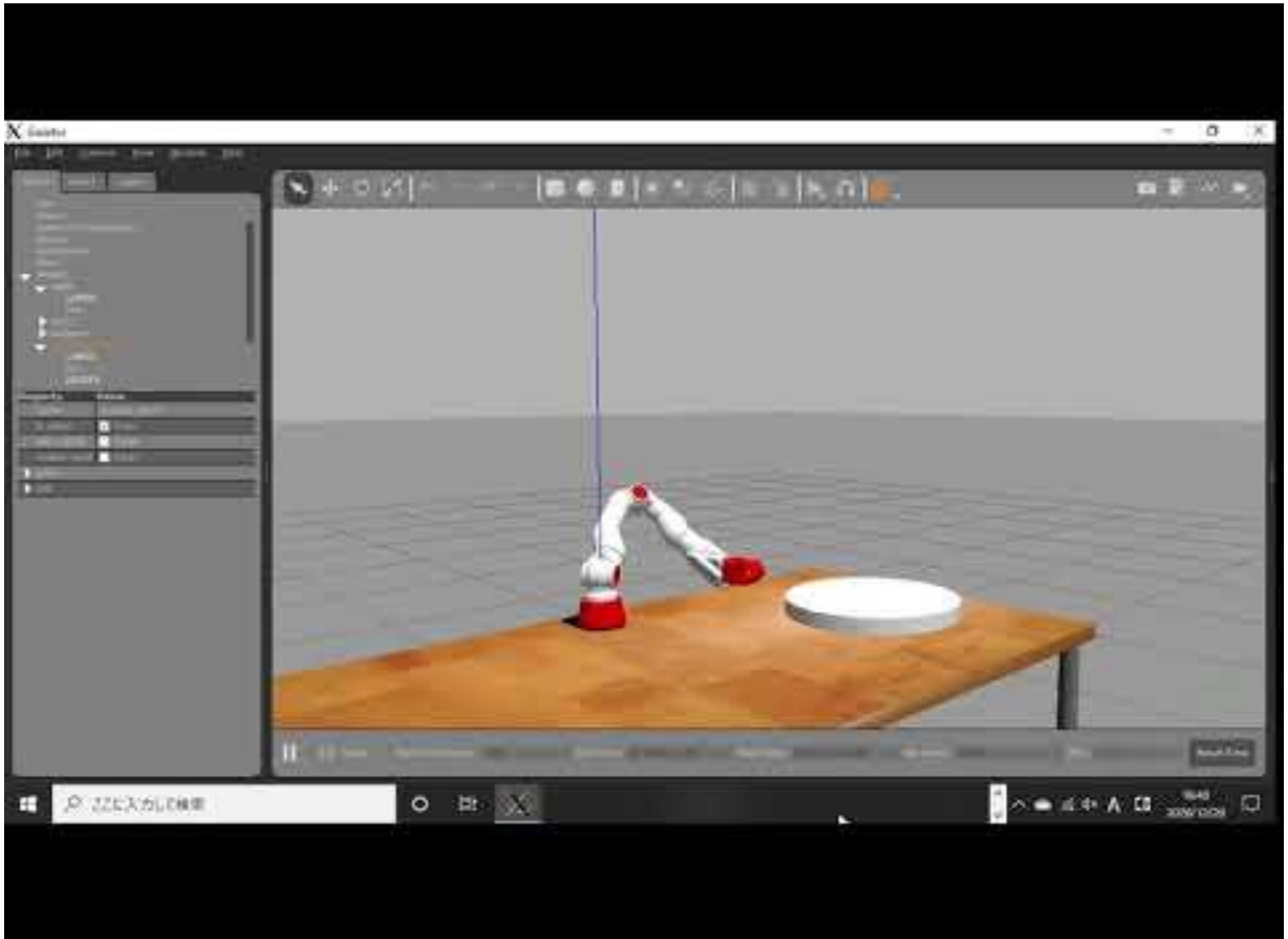
- タンバリンを叩くためのソースコード

アームを動かす関数の内で実機での調整が必要そうな部分は変数で置くことで、調整しやすくした。
タンバリンを叩く動作をfor文で回して動かすことで、プログラムが短くなるようにした。

```
# 何回叩くか
count = 3
# 叩いた後の持ち上げる高さ
adjustment = 0.02
# タンバリンの手前のx座標
tambourine_x_position_vertical = 0.043040
# タンバリンのy座標
tambourine_y_position_vertical = 0.303386
# タンバリンの少し上のz座標
tambourine_z_position_vertical = 0.085469
# 棒の角度
stick_angle_vertical = 1.3
```

```
for i in range(count):
    # タンバリンの真上にアームを移動
    preparation_vertical()
    # タンバリンを押す
    hit_tambourine_vertical()
    # タンバリンから離す
    preparation_vertical_up_little()
    rospy.sleep(0.1)
```

ソースコード解説 （中村）



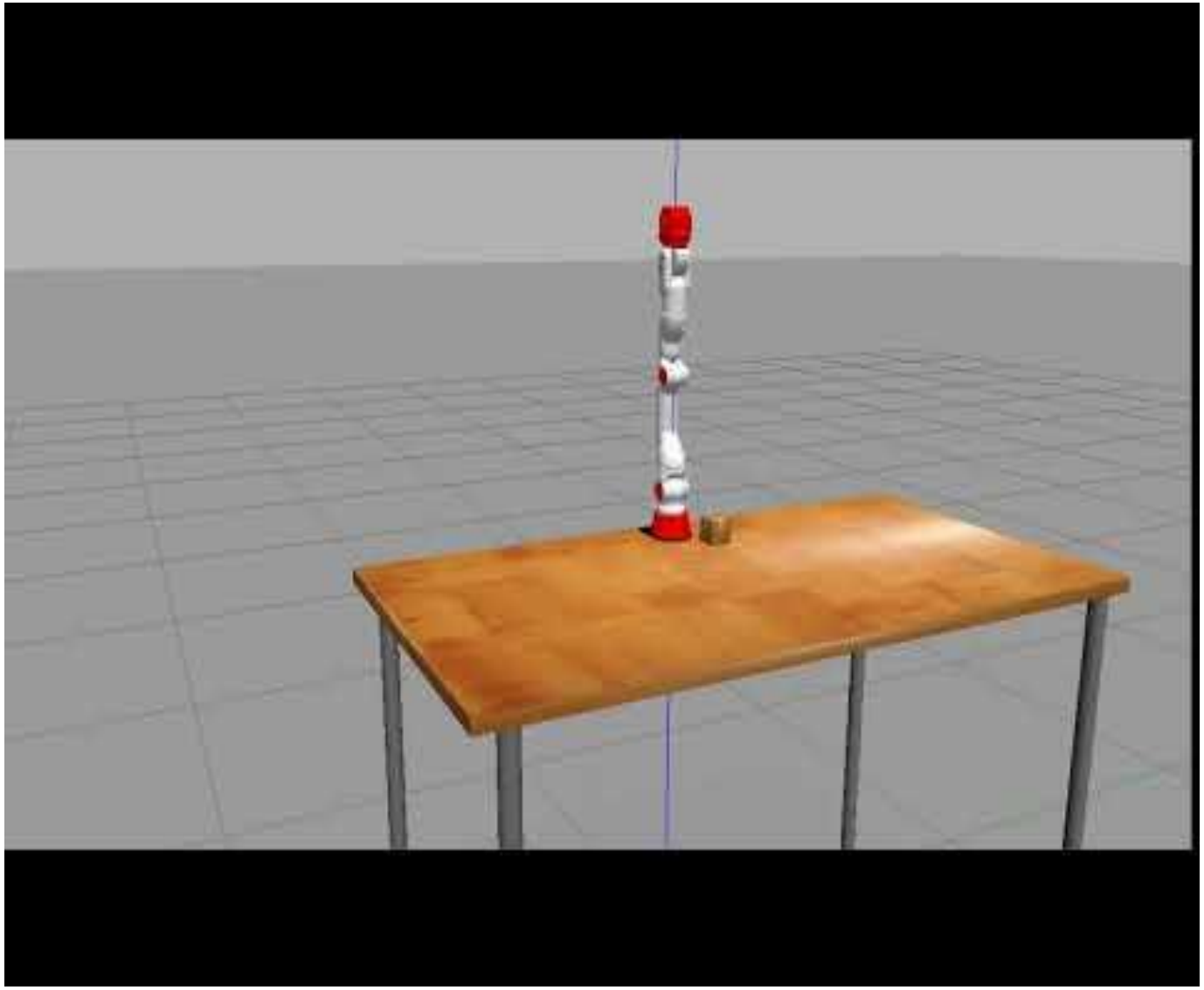
ソースコード解説（髙橋）

- マネキンを叩くためのソースコード

```
arm_x = 0.5
arm_y = 0.5
arm_z = 1.0

# SRDFに定義されている"vertical"の姿勢にする
# すべてのジョイントの目標角度が0度になる
arm.set_named_target("vertical")
arm.go()

# 目標角度と実際の角度を確認
print "joint_value_target (radians):"
print arm.get_joint_value_target()
print "current_joint_values (radians):"
print arm.get_current_joint_values()

# アーム初期ポーズを表示
arm_initial_pose = arm.get_current_pose().pose
print("Arm initial pose:")
print(arm_initial_pose)
```

```
#マネキンまで移動
def move():
    target_pose = geometry.msg.pose()
    target_pose.position.x = arm_x
    target_pose.position.y = arm_y
    target_pose.position.z = arm_z
    q = quaternion_from_euler( 0.0, 0.0, 0.0 )
    target_pose.orientation.x = q[0]
    target_pose.orientation.y = q[1]
    target_pose.orientation.z = q[2]
    target_pose.orientation.w = q[3]
    arm.set_pose_target(target_pose)
    arm.go()

# SRDFに定義されている"home"の姿勢にする
print("home")
arm.set_named_target("home")
arm.go()
```

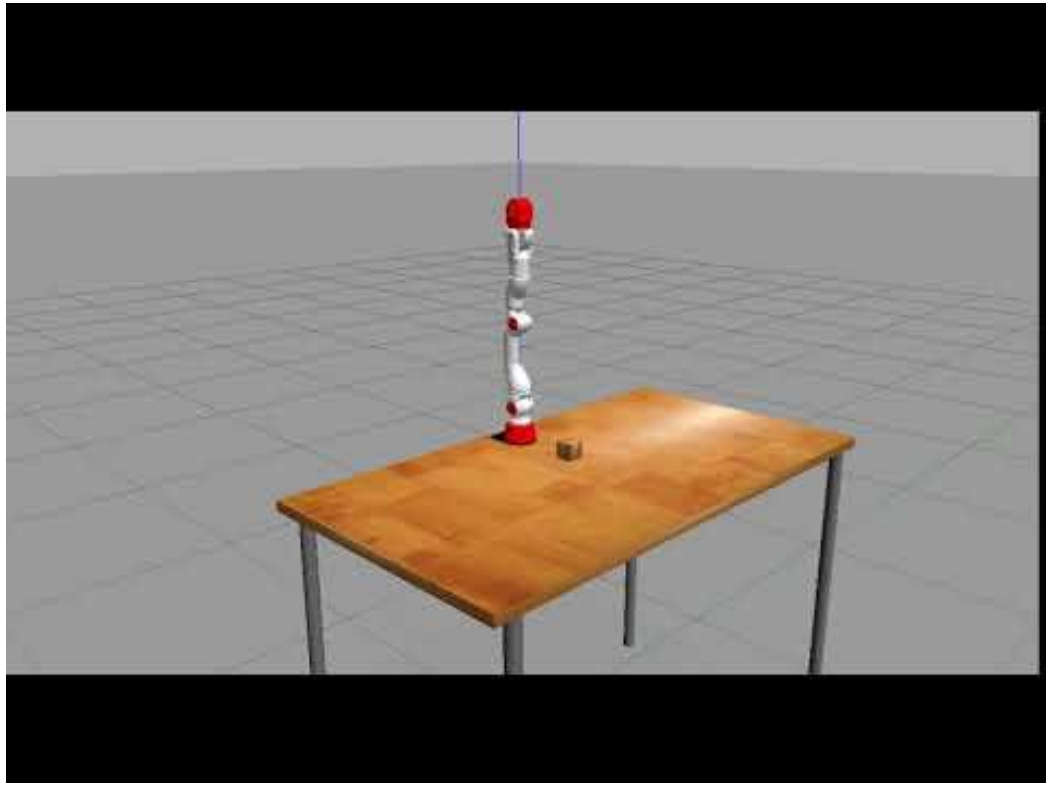
ソースコード解説（髙橋）



ソースコード解説（髙橋）

- マネキンを叩くためのソースコード（gazebo上のみ）

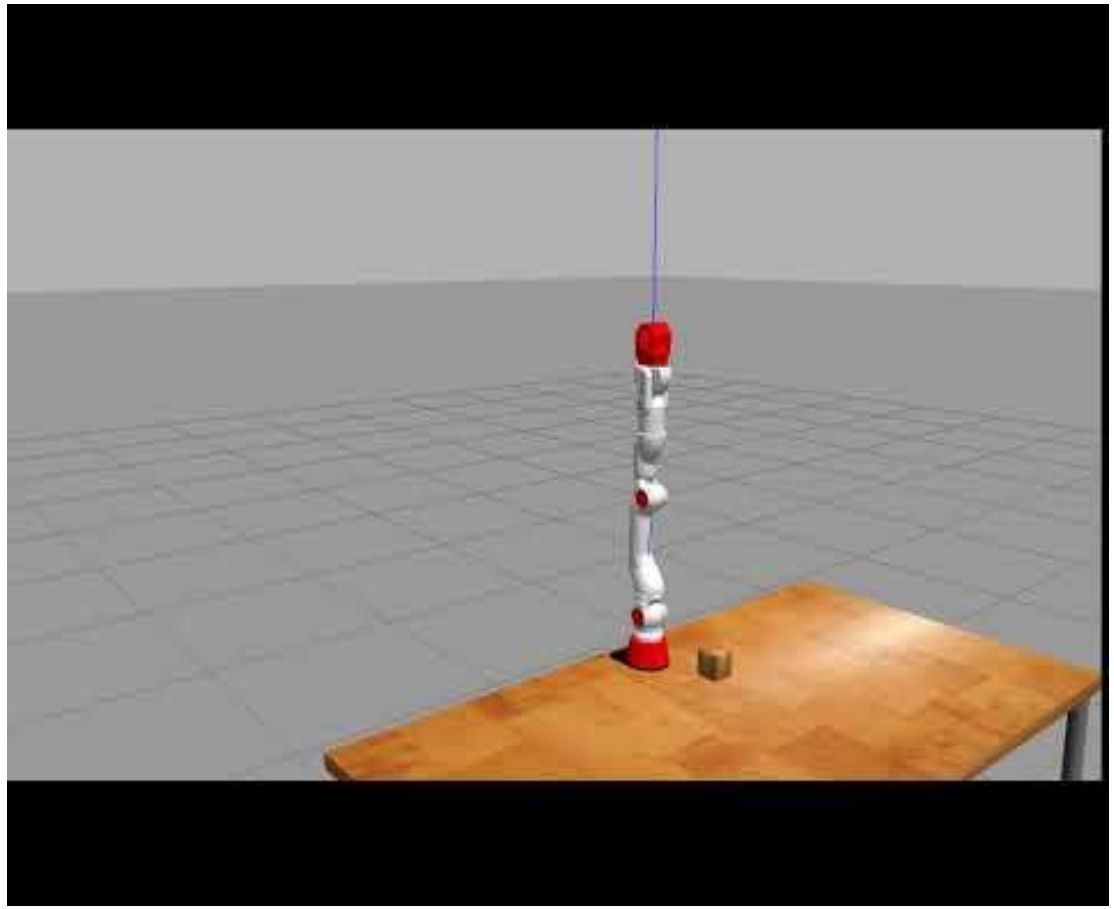
```
# 現在角度をベースに、目標角度を作成する
target_joint_values = arm.get_current_joint_values()

for num in range(3):
    joint_angle = math.radians(45)
    target_joint_values[5] = joint_angle
    arm.set_joint_value_target(target_joint_values)
    arm.go()

    joint_angle = math.radians(0)
    target_joint_values[5] = joint_angle
    arm.set_joint_value_target(target_joint_values)
    arm.go()


# SRDFに定義されている"vertical"の姿勢にする
print("vertical")
arm.set_named_target("vertical")
arm.go()
```



ソースコード解説（髙橋）

```
# 現在角度をベースに、目標角度を作成する
target_joint_values = arm.get_current_joint_values()
# 各ジョイントの角度を１つずつ変更する
joint_angle = math.radians(-90)
target_joint_values[2] = joint_angle

arm.set_joint_value_target(target_joint_values)
arm.go()
print str(2) + "-> joint_value_target (degrees):",
print math.degrees( arm.get_joint_value_target()[2] ),
print ", current_joint_values (degrees):",
print math.degrees( arm.get_current_joint_values()[2] )

joint_angle = math.radians(-90)
target_joint_values[3] = joint_angle
arm.set_joint_value_target(target_joint_values)
arm.go()

joint_angle = math.radians(90)
target_joint_values[6] = joint_angle
arm.set_joint_value_target(target_joint_values)
arm.go()

joint_angle = math.radians(90)
target_joint_values[2] = joint_angle
arm.set_joint_value_target(target_joint_values)
arm.go()


rospy.sleep(3)
```

- マネキンを叩くためのソースコード（gazebo上のみ）



ソースコード解説（熊谷）

・重要書類で脅す動き

動く座標設定

```
# アームを移動する
def move_arm(pos_x, pos_y, pos_z):
    target_pose = geometry_msgs.msg.Pose()
    target_pose.position.x = pos_x
    target_pose.position.y = pos_y
    target_pose.position.z = pos_z
    q = quaternion_from_euler(-3.14/2.5, 3.14, -3.14/2.0)
    target_pose.orientation.x = q[0]
    target_pose.orientation.y = q[1]
    target_pose.orientation.z = q[2]
    target_pose.orientation.w = q[3]
    arm.set_pose_target(target_pose)
    arm.go()
```

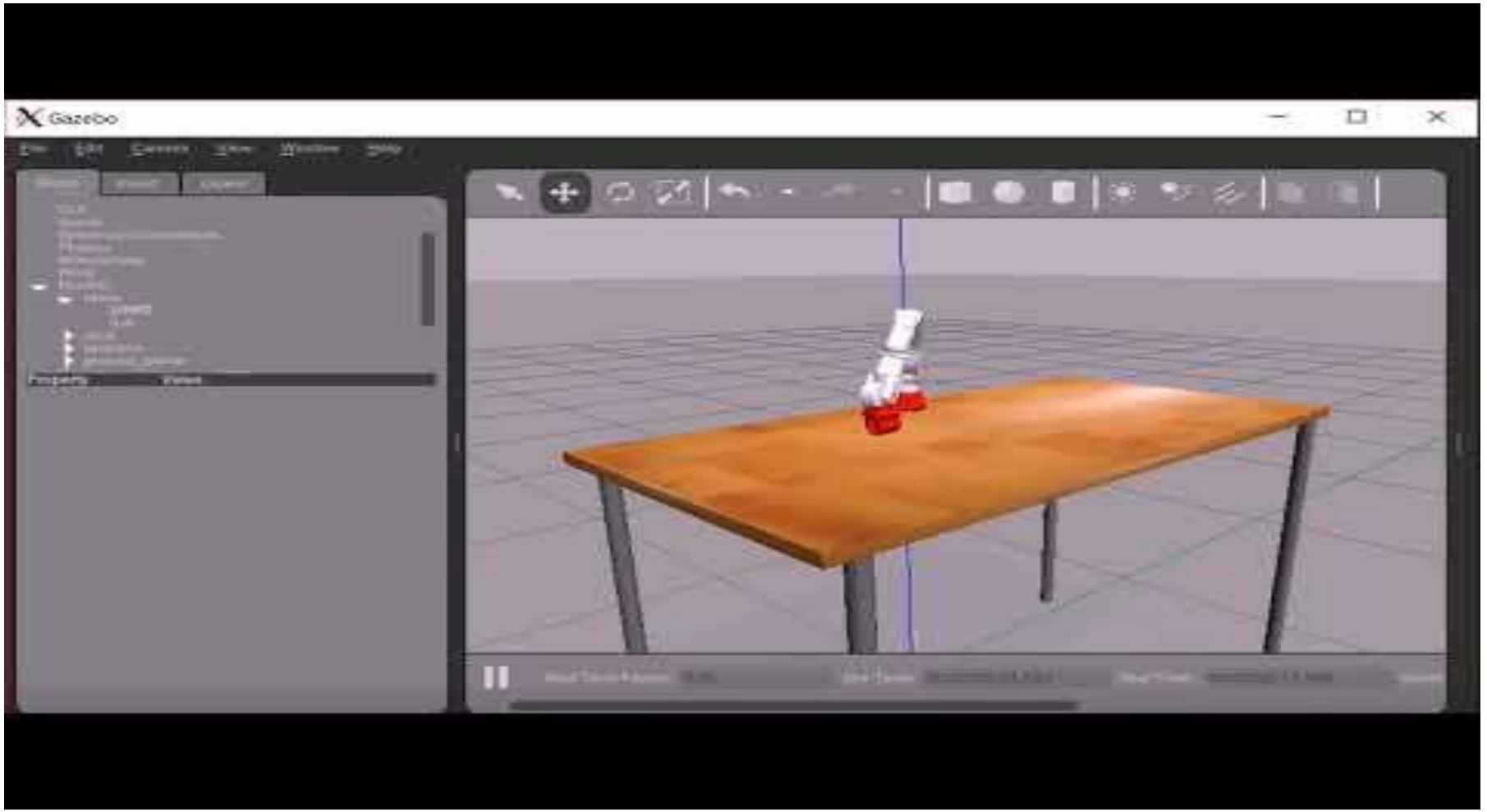
```
def preparation_vertical():
    target_pose = geometry_msgs.msg.Pose()
    target_pose.position.x = te_x_position_vertical
    target_pose.position.y = te_y_position_vertical
    target_pose.position.z = te_z_position_vertical
    q = quaternion_from_euler(-3.14/2.0, 3.14, -3.14/2.0)
    target_pose.orientation.x = q[0]
    target_pose.orientation.y = q[1]
    target_pose.orientation.z = q[2]
    target_pose.orientation.w = q[3]
    arm.set_pose_target(target_pose)
    arm.go()
```

動きの実行

```
preparation_vertical()
preparation_vertical()
p1_vertical()
preparation2_vertical()
p2_vertical()
preparation_vertical()
p3_vertical()
preparation2_vertical()
```

ソースコード解説 （熊谷）

gazebo



実演

結果

- 直接人を殴って起こす→嫌がらせに変更
- アームに持たせる物を統一化（ハリセン、ピコピコハンマー→棒）
- センサとアーム動作のプログラムの組み合わせが完成できた。
- 各々の作業を予定通り完了させることができた。

課題点

・目の誤検出が多かった
・センサに対して正面を向かなければ検出ができない。

・gazebo上で簡易的なシミュレーションしかしなかったため
　実機で調整がする部分が多くなってしまった。

各自の作業

- 熊谷　和真：紙をつっつく動作の作成
- 酒谷　冬芽：各自が作成したものを組み合わせる　動作確認
- 髙橋　祐樹：マネキンを叩く動作の作成
- ホンスワン　タナポル：　センサの実験
- 中村　亮：　タンバリンのモデルと叩く動作の作成

リンク

- 作成したRosパッケージ

- 実演動画

ご清聴ありがとうございました