エージェントシステム

稲葉研

（垣内）

2019/4/24

課題

Choreonoidで振付をする

- 人型のロボットの動作（踊り、挨拶、仕事など）の振付をする
  - 最終動作の画面をキャプチャ
  - どういう機能を用いて振付をしたか記述

（アドバンス：バランスを考慮してみる）

（アドバンス：独自のロボットモデルをつくってみる）

JVRC-1

https://github.com/jvrc/model/tree/master/JVRC-1

JAXON

https://github.com/start-jsk/rtmros_choreonoid/tree/master/jvrc_models/JAXON_JVRC

課題

Dockerでシミュレーションを動かす

- WRS2018のタスクのどれかを実行
  - スタート地点からできるところまでロボットを動かす
  - 操作の画面をキャプチャ

  （アドバンス: カメラ画像のを見て操作する）

- プログラムを修正する
  - ジョイスティック以外で動かしてみる
  - 自律動作を作ってみる

  （ヒント: /joy に何かトピックを与えると動くだろう）

課題

kyoin@jsk.t.u-tokyo.ac.jp

Subject: エージェントシステム課題1回目

本文に　所属専攻 研究室 学生証番号 氏名
を記述して、pdfを添付で上記メールアドレスへ
適宜ビデオ等も添付のこと

締め切り:　５月８日講義まで

仮想化環境

- 仮想化環境の種類
- エミュレータ/異種類のCPUのコードを実行
  - Qemu（AndroidSDK）
- アプリケーション/アプリとしてバイトコード実行
  - Java (JavaVM)
- PCレベル仮想化/OSを1からインストール
  - VirtualPC, VMware
- OSレベルの仮想化/カーネルの共有
  - Docker

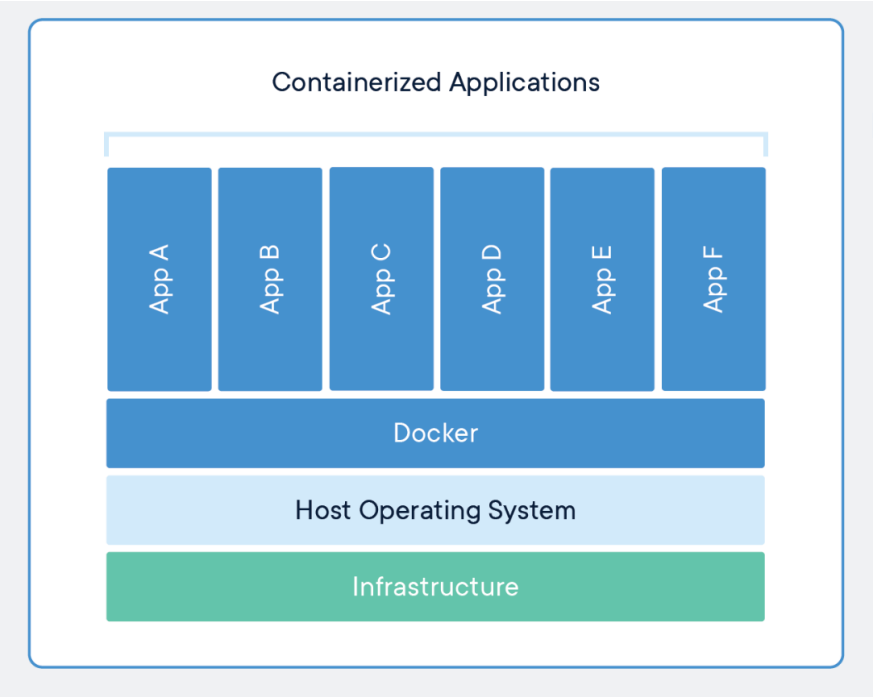
仮想すると良いこと

- なぜ仮想化するのか
- 異なるアーキテクチャのCPUを使いたい
- 複数のOSを使いたい（デュアルブート）
- 異なる環境（CPU，OS，バージョン）で同じ結果になって欲しい
- ポータブル（移動が可能に）
- カプセル化
  - ブラックボックス（セキュリティ）
  - 変更が簡単に戻せる

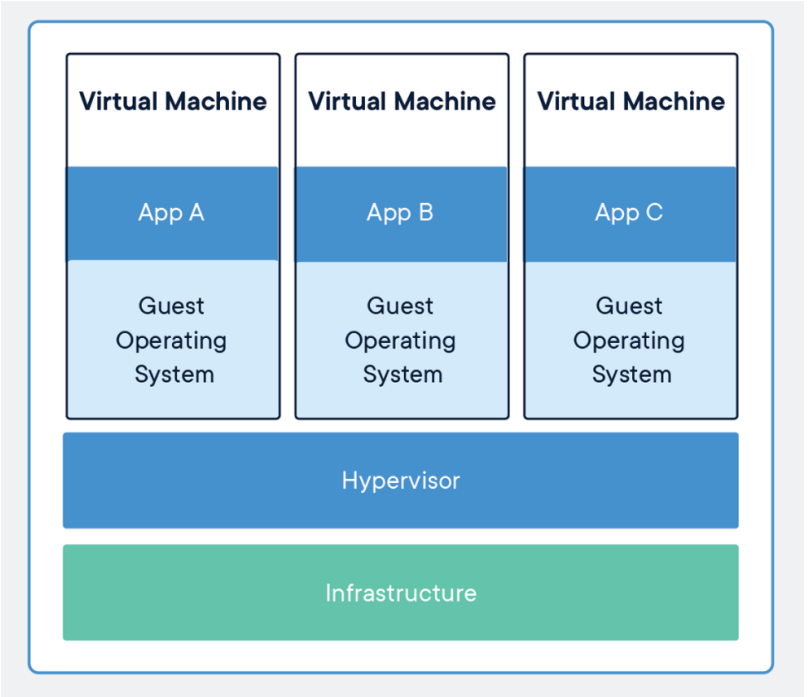
Dockerとは

- ホストOSのカーネルを共有
- OSレベルのインストールが必要なくなる
- イメージはレイヤで管理されレイヤは共通化される

Docker



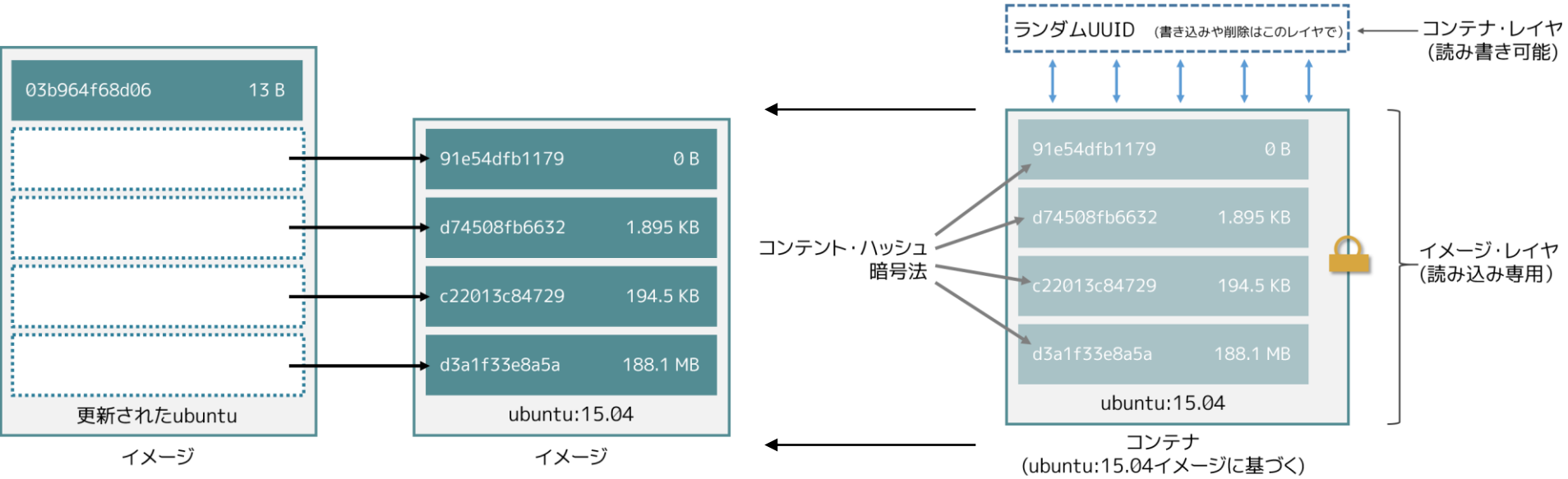
VMwareなど



https://www.docker.com/resources/what-container

Dockerとは

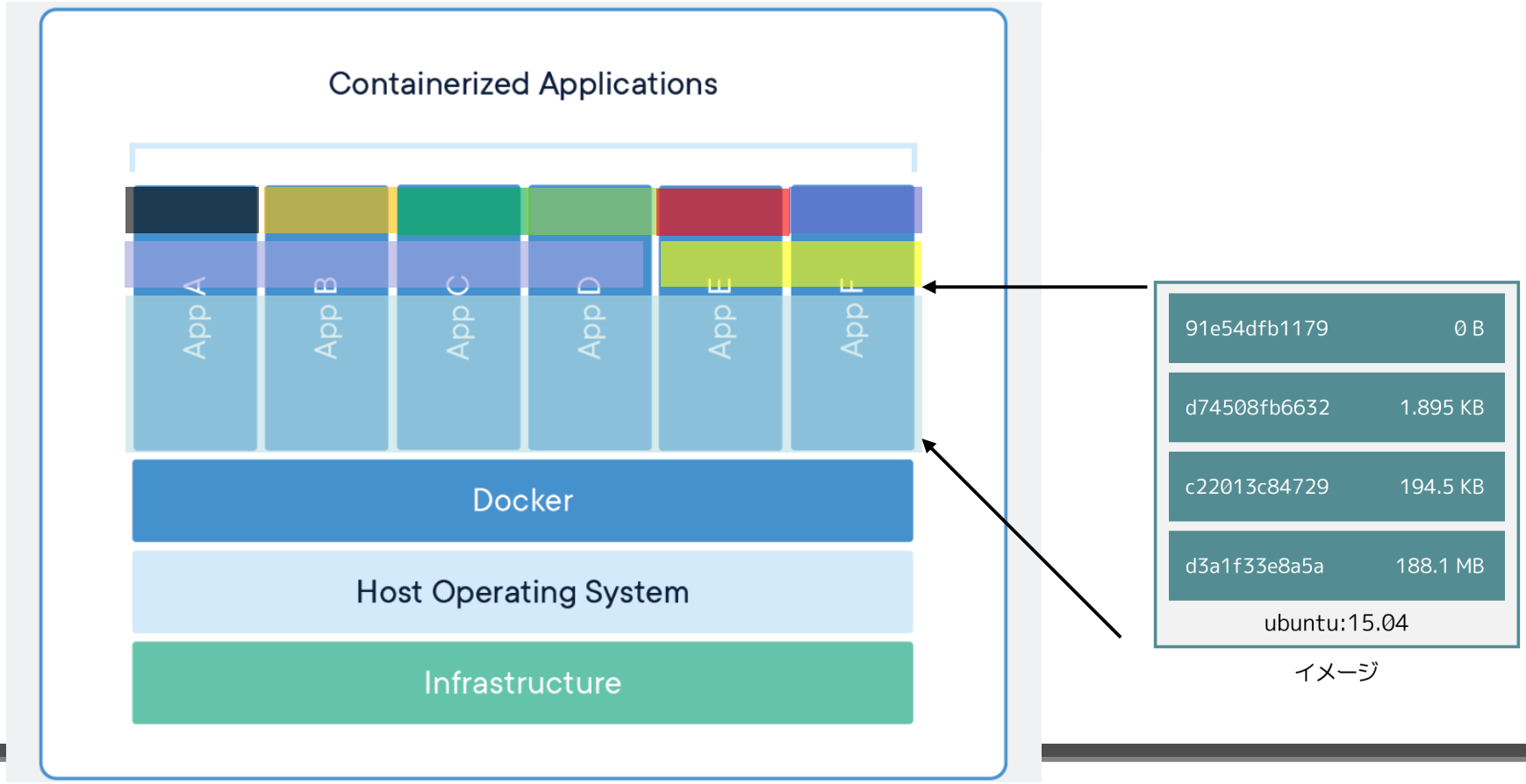
- ホストOSのカーネルを共有
- OSレベルのインストールが必要なくなる
- イメージはレイヤで管理されレイヤは共通化される



http://docs.docker.jp/engine/userguide/storagedriver/imagesandcontainers.html

Dockerとは

- ホストOSのカーネルを共有
- OSレベルのインストールが必要なくなる
- イメージはレイヤで管理されレイヤは共通化される



Dockerhub

- イメージを共有（公開）するしくみ
- https://hub.docker.com/
- 多くのオフィシャルイメージ
  - OS (Ubuntu, Debian, CentOS, …)
  - 様々なアプリケーション (httpd, SQL, …)
  - ホストコンピュータの環境を変えずに試せる、運用できる
  - ソフトウェアが更新されるとイメージも自動的に更新される

Dockerhub

- GPUも使える Nvidia-docker
- https://github.com/NVIDIA/nvidia-docker

- DeepLerning環境もインストール簡単
- chainer
- https://hub.docker.com/r/chainer/chainer/
- tensorflow
- https://hub.docker.com/r/tensorflow/tensorflow/

Dockerのインストール

- YoheiKakiuchi/robotsimulation-docker
  - https://github.com/YoheiKakiuchi/robotsimulation-docker
  - READMEを読んでnvidia-docker2をインストール(NVIDIAユーザー)
  - NVIDIAでない人はdockerまで
  - gitのレポジトリをクローンする
  - git clone https://github.com/YoheiKakiuchi/robotsimulation-docker.git

- robotsimulation-docker/choreonoid_docker
  - dockerのイメージをpullしてきてデモプログラムを試す
  - NVIDIA
  - docker pull yoheikakiuchi/choreonoid:16.04_latest_ros
  - Not NVIDIA
  - docker pull yoheikakiuchi/choreonoid:16.04_no_gl_latest_ros

Dockerの使い方

```
$ docker run –it ubuntu:xenial bash
$ docker run –it ubuntu:bionic bash

$ docker run –it osrf/ros:kinetic-desktop bash
$ docker run –it osrf/ros:melodic-desktop bash

$ roscore
$ docker run --net=host osrf/ros:melodic-desktop
rostopic pub –r 1 /hoge std_msgs/String
‘docker_test’
$ docker run --net=host osrf/ros:kinetic-desktop
rostopic echo /hoge
```

Dockerの使い方

docker runでコンテナの作成

```
$ docker run –it --name=test_container
osrf/ros:kinetic-desktop bash
```

docker execで動いているコンテナでプロセス実行

```
$ docker exec –it test_container bash
```

コンテナの一覧

```
$ docker ps –a
```

イメージの一覧

```
$ docker images
```

Dockerの使い方

```
$ docker images
$ docker ps –a
```

```
$ docker run –it ubuntu:xenial bash
```

(何回runしてもイメージは変わらない)

```
$ docker commit (container name) (image name)
```

(コミットすると修正したコンテナのイメージが作られる)

```
$ docker run –it (image name) bash
```

(コミットしたイメージの実行)

Choreonoid

- 産総研 中岡慎一郎が主な開発者
- 公式サイト: https://choreonoid.org/ja/
- Windows, Linuxで動作
- オープンソースソフトウェア（MITライセンス）
- https://github.com/s-nakaoka/choreonoid
- ロボットモデルを扱う基本機能、動力学シミュレーション機能を内蔵
- ユーザがプラグインを開発することにより、様々な機能拡張が可能
- シングルプロセス型の実装で軽快に動作

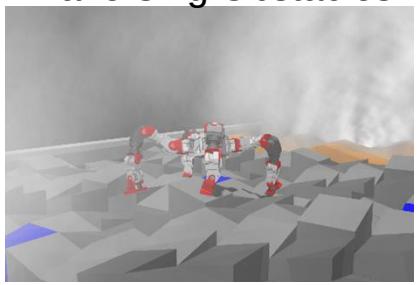
ワールドロボットサミット

World Robot Summit

インフラ・災害対応カテゴリ／トンネル事故災害対応・復旧チャレンジを対象として、Choreonoidを用いたシミュレーション競技を開催（2018年10月）

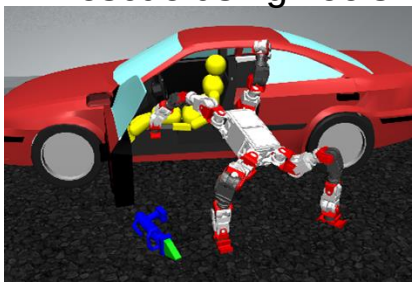
2020年に実機で開催予定？

Task T1
Traversing Obstacles

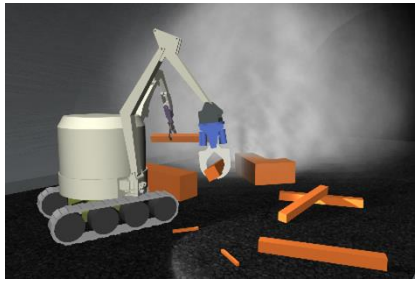


Task T2
Vehicle Inspection

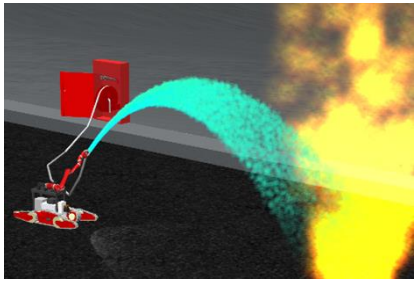
Task T3
Rescue using Tools



Task T4
Secure the Route



Task T5
Fire Extinguish



Task T6
Shoring and Breaching

Choreonoid on Docker

- robotsimulation-docker/choreonoid_docker
  - dockerのイメージをpullしてきてデモプログラムを試す
  - NVIDIA
  - docker pull yoheikakiuchi/choreonoid:16.04_latest_ros
  - Not NVIDIA
  - docker pull yoheikakiuchi/choreonoid:16.04_no_gl_latest_ros

  /userdir に Current_directory (run.shを実行した robotsimulation-docker/choreonoid_docker ディレクトリ) がマウントされる

  run.sh を実行したディレクトリにdocker側に渡したいファイルを置いておく

Choreonoid on Docker

- YoheiKakiuchi/robotsimulation-docker
  - https://github.com/YoheiKakiuchi/robotsimulation-docker
  - READMEを読んでnvidia-docker2をインストール(NVIDIAユーザー)
  - NVIDIAでない人はdockerまで
  - gitのレポジトリをクローンする
  - git clone https://github.com/YoheiKakiuchi/robotsimulation-docker.git

- robotsimulation-docker/choreonoid_docker
  - dockerのイメージをpullしてきてデモプログラムを試す
  - NVIDIA
  - docker pull yoheikakiuchi/choreonoid:16.04_latest_ros
  - Not NVIDIA
  - docker pull yoheikakiuchi/choreonoid:16.04_no_gl_latest_ros

Choreonoidによる振付

チュートリアル

- https://choreonoid.org/ja/tutorials.html

ロボットモデルから作る

- File -> Open Project -> SR1Minimum.cnoid
  Samplerobot minimumを選択
- File -> New -> PoseSeq
- View -> Show View -> Pose Roll
- Pose Roll
  - Insert
  - Update

Choreonoidのシミュレーション (WRS2018)

WRS2018 チュートリアル

- https://choreonoid.org/ja/manuals/latest/wrs2018/index.html
- robotsimulation-docker/choreonoid-docker にて

$ roscore ### ノートパソコンなど ./run_choreonoid roscore

$ DOCKER_OPTION=-it ./run_choreonoid.sh bash

$ choreonoid /choreonoid_ws/devel/share/choreonoid-1.7/WRS2018/script/T1M-AizuSpiderSS-ROS.py

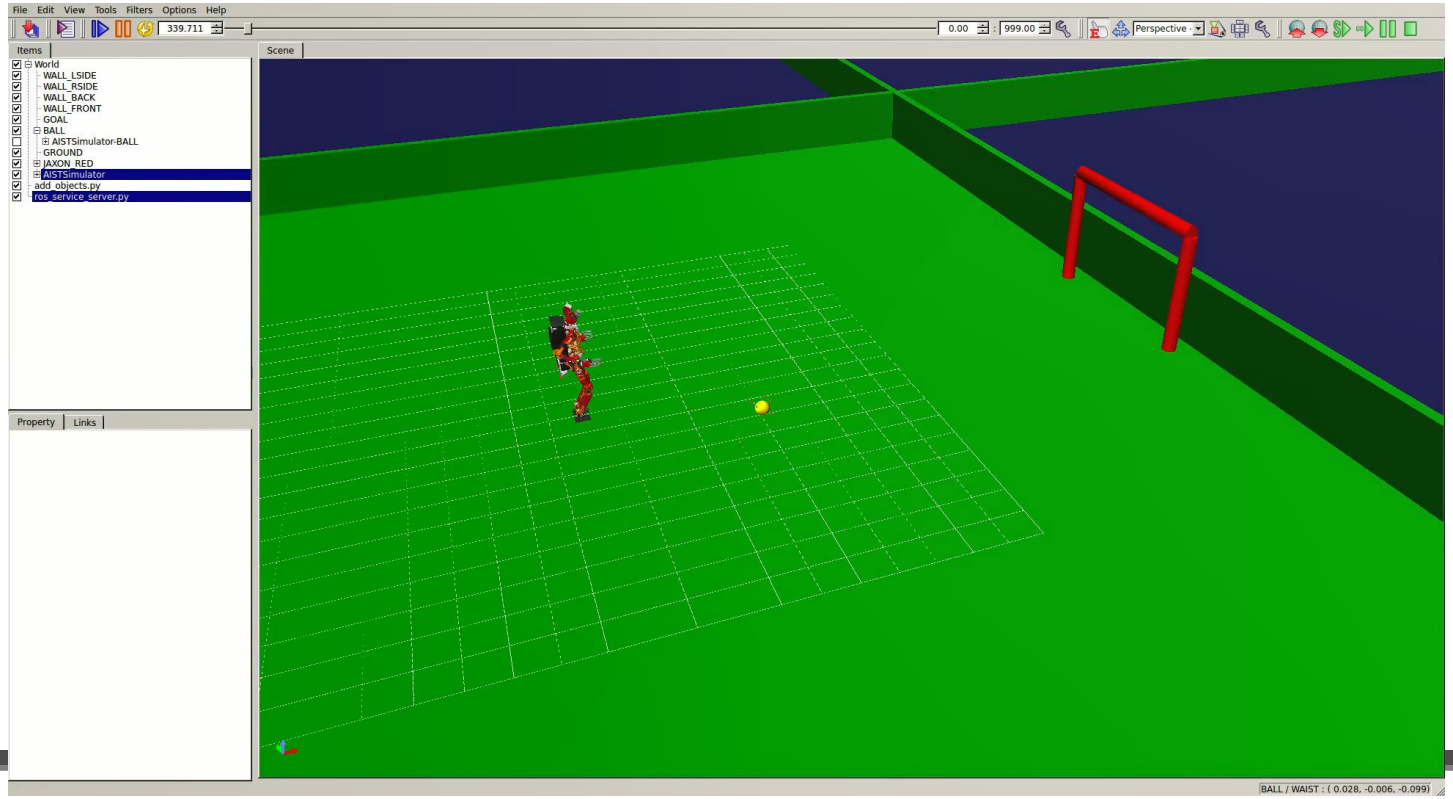
$ ./exec_choreonoid.sh /my_entrypoint.sh rosrun choreonoid_joy node

ヒューマノイドロボット choreonoid_docker

- YoheiKakiuchi/robotsimulation-docker
  - https://github.com/YoheiKakiuchi/robotsimulation-docker
  - READMEを読んでnvidia-docker2をインストール(NVIDIAユーザー)
  - NVIDIAでない人はdockerまで
  - gitのレポジトリをクローンする
  - git clone https://github.com/YoheiKakiuchi/robotsimulation-docker.git
- robotsimulation-docker/choreonoid_docker
  - dockerのイメージをpullしてきてデモプログラムを試す
  - NVIDIA
  - docker pull yoheikakiuchi/choreonoidsim:16.04_release-1.6
  - Not NVIDIA
  - docker pull yoheikakiuchi/choreonoidsim:16.04_no_gl_release-1.6

ヒューマノイドロボット choreonoid_docker

- robotsimulation-docker/choreonoid_docker
  - choreonoid_docker/README.md の以下の表題のコマンドを試す
    - Run simulation (1)
    - Run perception nodes (2)
    - Run robot kicking ball (run with perception nodes) (3)



ユーザープログラムの試し方

- サンプルプログラム
  - robotsimulation-docker/choreonoid_docker/sample-program.lを変更することでプログラムが試せる
  - ./run.sh で choreonoidを立ち上げる
  - ./exec.sh '/my_entrypoint.sh roseus /userdir/user_programs/sample-program.l (your-function)‘
  - choreonoid_docker/README.md (Runnig your programs)に同じことが書いてある
  - もしインタプリタでeuslispが使いたければ，以下のように (emacs内の*shell*などで立ち上げると使いやすい，emacs にて M-x shell）
  - ./exec.sh ‘/my_entrypoint.sh roseus‘

ユーザープログラムの解説

- (require :jvrc-standup "package://hrpsys_choreonoid_tutorials/euslisp/action_and_perception/jvrc-statenet.l")
- (jaxon_jvrc-init) ;; ロボット モデルの作成とシミュレーションとの通信の確立
- ;; you should comment out for walking ;; 初期姿勢を作る
- (send *ri* :stop-impedance :arms)
- (send *robot* :reset-pose)
- (send *robot* :fix-leg-to-coords (make-coords))
- (send *robot* :move-centroid-on-foot :both (list :rleg :lleg))
- (send *ri* :angle-vector (send *robot* :angle-vector) 1000)
- (send *ri* :stop-st)
- (send *ri* :stop-auto-balancer)
- ;;

ユーザープログラムの解説

```
(defun your-function () ;; 関数の定義 ./exec.sh において呼んでいる
  ;; robot is moved to face-up posture
  (reset-position :coords (make-coords
                           :pos (float-vector 0 0 300) ;; (x y z) [mm] position of waist
                           :rpy (list 0 -pi/2 0) ;; (yaw pitch roll) [rad] rotation of waist
                           ))  ;; ロボットを仰向けに寝かせる（シミュレーション内のロボットを強制的に）
  (objects (list *robot*)) ;; ロボットを表示する
  (send *robot* :reset-pose) ;; ロボット（モデル）を初期姿勢にする
  ;; the robot in the simulation moves to the same pose as *robot*. it takes base-time [ms].
  ;; シミュレーション内のロボットにモデルと同じ姿勢となるように動作指令を送る
  (send-pose :real t :base-time 2000)
  ;; 関数の実行
  (face-up-to-face-down-action)
  )
```

ユーザープログラムの解説

- ;; sample of face-up-to-face-down motion
- ;; 呼ばれたプログラムは https://github.com/start-jsk/rtmros_choreonoid/blob/master/hrpsys_choreonoid_tutorials/euslisp/action_and_perception/jvrc-standup.l#L260-L301 に書いてある
- (defun face-up-to-face-down-action (&key (real nil) (base-time 2500))
- ;; pose 0
- (なんらかの姿勢を変える命令）
- (send-pose :real real :base-time base-time) ;; ロボットを動かす

- ;; pose1 ~ pose5 までそれの繰り返しのプログラムである
- ）

参考資料

- https://github.com/euslisp/jskeus/blob/master/doc/jmanual.pdf 第III部 irteus拡張を参照
- http://wiki.ros.org/rtmros_common/Tutorials/WorkingWithEusLisp
- http://euslisp-tutorial.readthedocs.io/ja/latest/robot_coding/（JSK 矢口先生）