オンラインでの
組込み教育・モデリング教育について
講演資料

久保秋　真
（株）チェンジビジョン

オンラインでの組込み教育・モデリング教育について

講演資料

久保秋 真

バージョン pdf_0115, 2021-09-03

目次

はじめに

このセッションは、オンラインでの組込み教育・モデリング教育について、わたしの実際に取り組んでいる方法などを紹介します。

お願い

このセッションでは、後半に実施する演習で GitとGitHub そして PowerPoint を使います。
いっしょに演習をやってみたい人は、次のことを済ませておいてください。

- Gitコマンドのインストール
- GitHubのアカウントの作成
- PowerPoint（MS Office）のインストール

また、演習で作成する文書のプレビューに、Asciidoctor のブラウザ用機能拡張を使います。次のページから、自分が使っているブラウザの機能拡張をインストールしておいてください。

Asciidoctor Browser Extension

https://github.com/asciidoctor/asciidoctor-browser-extension

1. 組込みモデリング教育のポイント

対面かオンラインかに依らず、教育でねらっていること。

1.1 講義、研修の実績

ソフトウェア設計、開発、オブジェクト指向、モデリングなどについて、講義や研修を担当。

大学、大学院

- 日本大、情報系学科3年生、クォーター:週2コマ×5.5回の講義(2006～)(図 1.1)
- 関東学院大、情報系学科2年生、半期:週2コマ×15回(2019～)
- 早稲田大、情報系大学院1年生、半期:週1コマ×15回(2009～)
- はこだて未来大、情報系学科3,4年生、スポットの2コマ1回(2009～)
- ダルマ・プルサダ大(ジャカルタ)、2,3年生4コマから6コマ(2018～)(図 1.2)
- enPIT-embでの演習、講義、MDDロボットチャレンジ(2004～)

社会人向け

- 企業の新人教育や中堅研修、多数実施、2日から7日(2004～)
- ETロボコン本部審査員、技術教育(2006～)
- トップ・エスイー(2009～)集中講義2日
- スマート・エスイー(2018～)、2コマ×2日

1.2 対面教育の素材

使用する機器

- Mindstorms EV3/NXT/RCX
- Raspberry Pi 3,4,Mini
- iRobot Create

モデリング演習

- 模造紙＋ポストイット
- モデリングツール
  - astah*
  - Enterprise Architect
  - BridgePoint

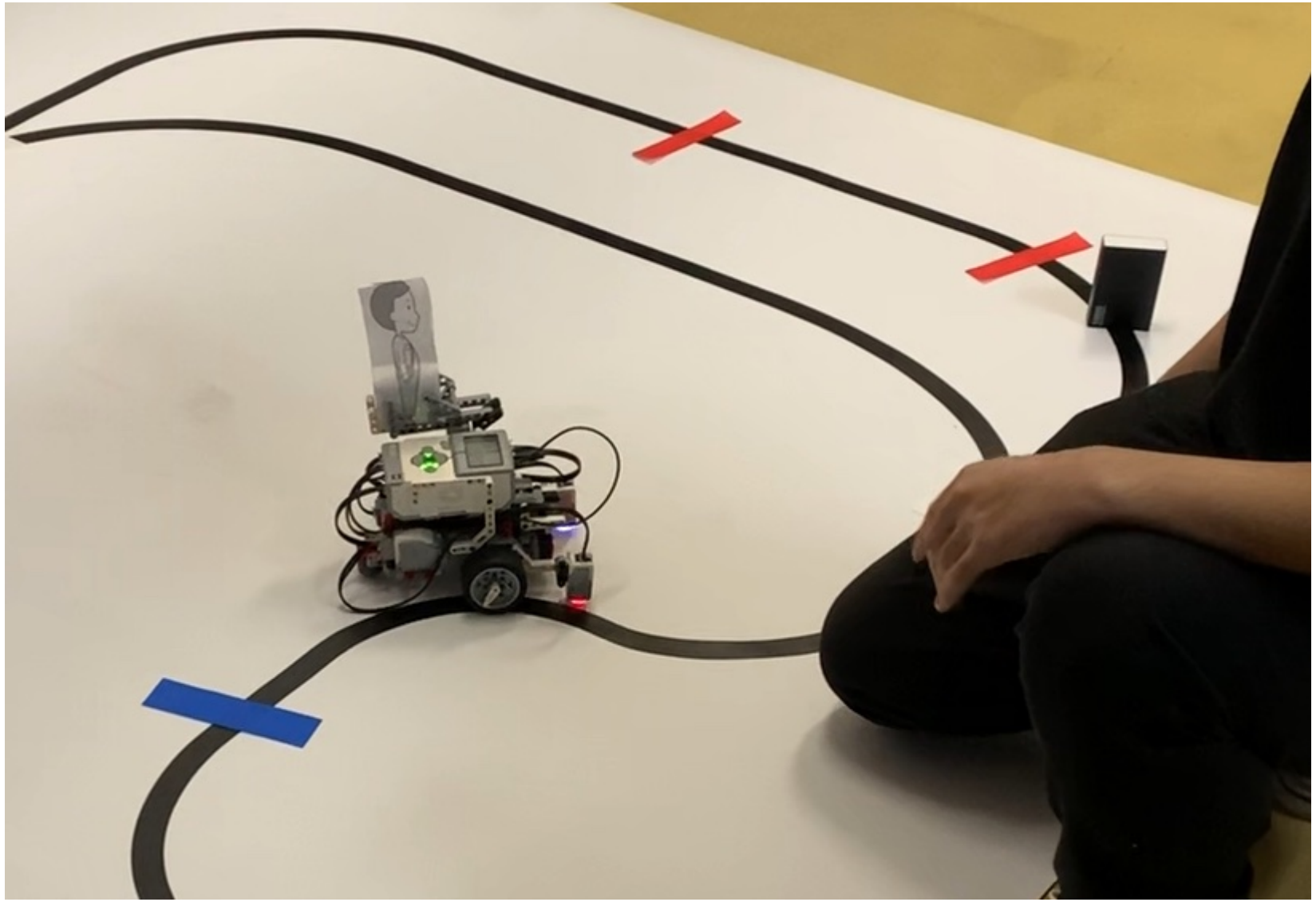


図 1.1 日本大学生産工学部情報数理学科の演習の様子

図 1.2 ダルマ・プルサダ大での演習の様子

図 1.3 MDDロボットChallengeの様子1(2009)

図 1.4 MDDロボットChallengeの様子2(2009)

図 1.5 MDDロボットChallengeの様子3(2014)

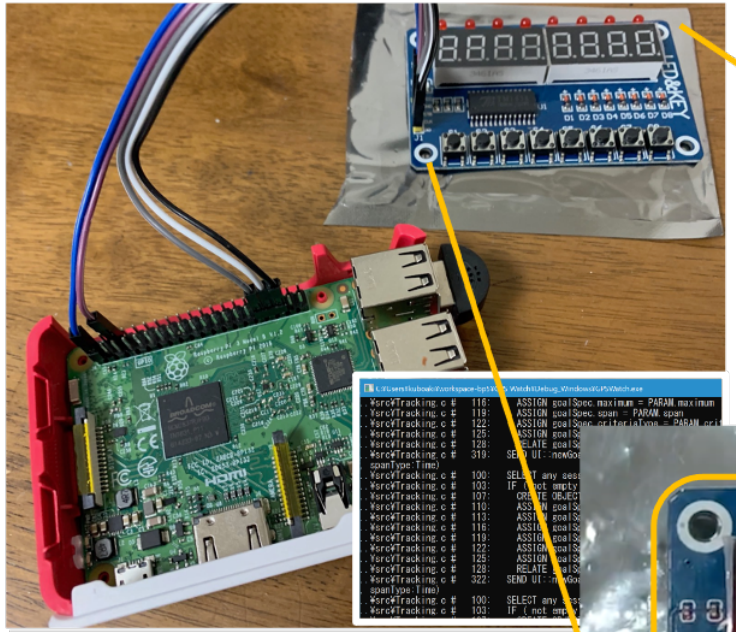
PC

水道局センターの
情報表示・操作
コンソール
（Java GUIアプリ）

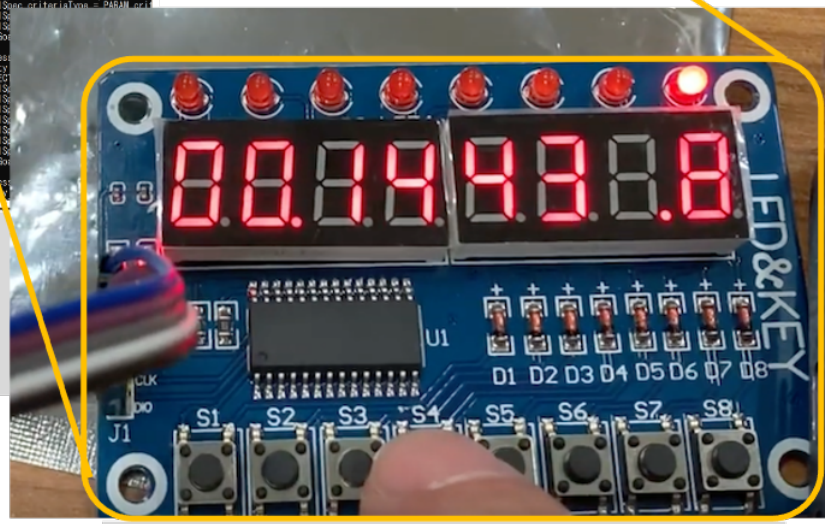
表示データ更新通知

TCP/IP経由
で通信

ボタン操作イベント



スマート水道メーターの
コアデバイス
（Raspberry Pi）



スマート水道メーター本体の
情報表示・操作画面
（TM1638つき基板）

図 1.6 スマート・エスイーの演習の例（スマート水道メーターに見立てた）

1.3 教育のねらい

- 設計の有用性を実感する
- 組込みでモデルを使い、モデルから実装へ
- 設計と実装の間に乖離の生まれる理由に気づく
- モデル駆動開発へ誘う

オンラインになっても、これらは変わらない。変えずに済ませるには、どうすればよいか。

1.4 ねらいに集中する

- モデルとコードの対応づけを実習する(図 1.7)
- 変換による実装を使う
  - モデルからコードへの変換ルールを決めて、あえて手動変換で実装する
- 変換による実装を前提に設計モデルを作る

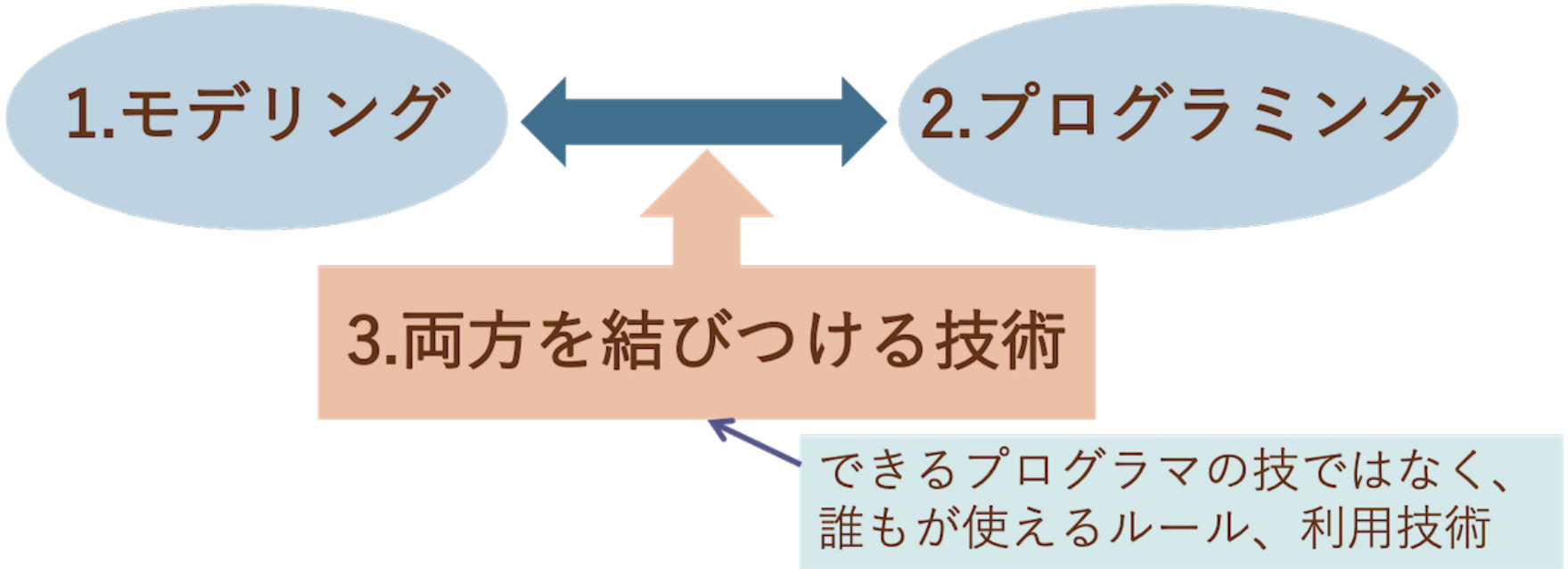


図 1.7 モデリングとプログラミングを結びつける技術

1.5 他のことについては限定的に

- UML、モデル、RTOSなどについて講義しない
  - 使う場面や必要な場面で説明する
- UMLは、クラス図、ステートマシン図、パッケージだけ
  - 記法は細かく説明しない
  - モデルを作る手順（要素の探し方など）や、構成方法を演習しながら教える
  - ただし、**クラス名、メソッド名、イベント名、アクション名にはこだわる**
- 言語はCかC++
  - IDEはなし（最近は、みんなVS Codeを使っている）
  - 難しい構文やスタイルは使わない
- OSはTOPPERS（EV3RT,nxtOSEK）、またはRaspbian
  - タスクは最低限、イベントキューなしでポーリングで済ませる、フラグやセマフォなしで状態で賄う
  - 高度な演習は、中堅向けや、演習する期間が長いときに
    - RTOSのサービスをモデルに閉じ込められるようになったら

1.6 最初は勝手に試行錯誤させない

モデル作成から実装まで、成果物がつながるようになったら、応用課題やグループ演習へ進む。

- やり方が身につくまで
  - まずはやらせてみるとか、できたら発表といった方式はとらない
  - できかけをプロジェクターで共有、講師や周囲と一緒に作ってやり方を教え込む
  - 周囲は、発表者のモデルやコードができていく様子をみて、まねる
- やり方が身についたら
  - 応用課題に取り組む、グループで取り組む
  - やり方を変えたら、動作しても評価されない
    - 関東学院大の成果物の評価ポイント(図 1.8)

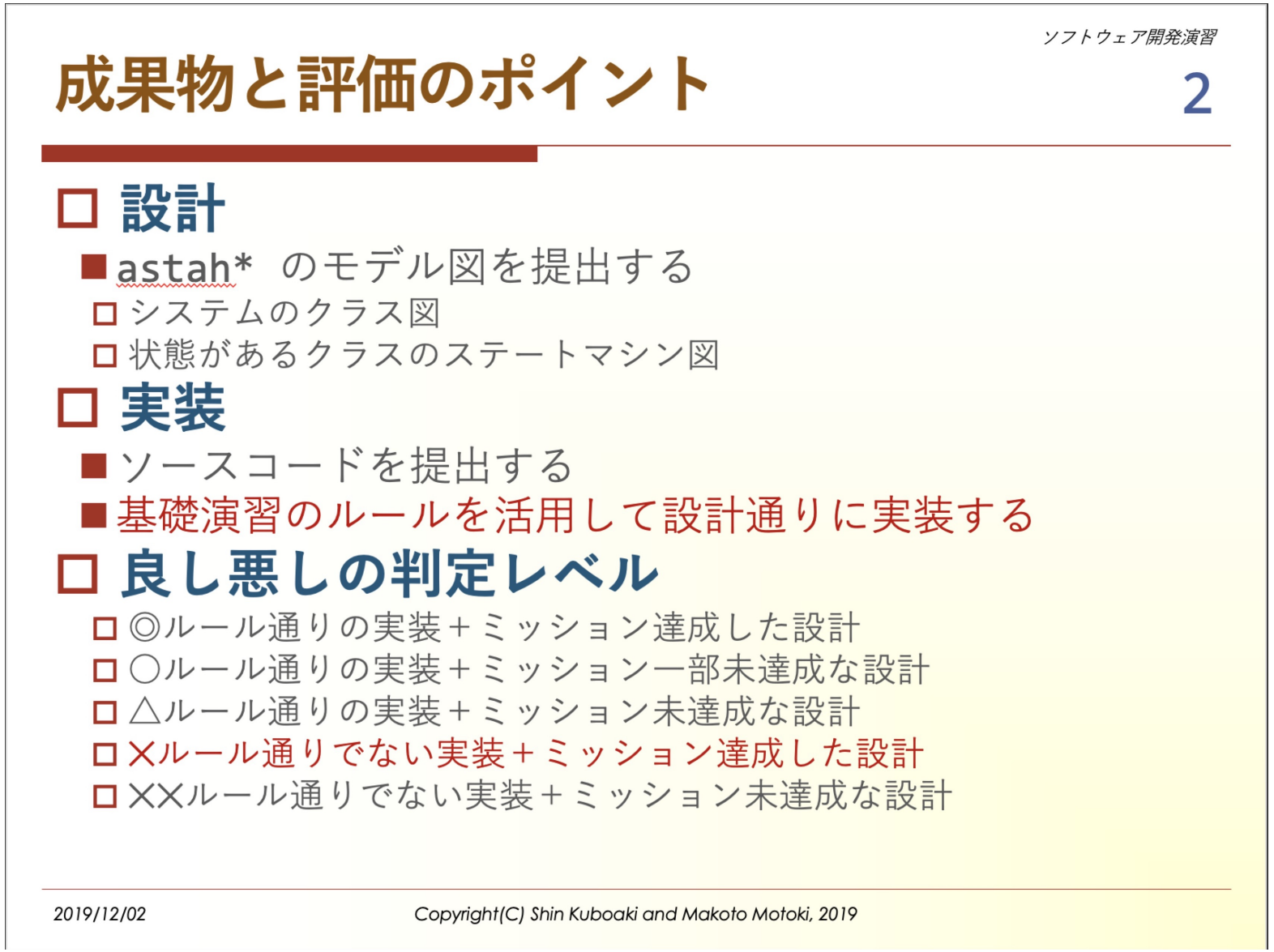


図 1.8 関東学院大での最終課題の評定基準

2. 演習のオンライン対応

対面では難しいところを、どのように対処したか。

2.1 機器を動かす演習（対面）

- 実機用の開発環境の構築
- EV3RT（NXTではnxtOSEK）のサンプル・プログラムを実行する
  - プログラムのビルド手順の把握
  - センサーやモーター等のAPIの使い方を知る
  - プログラム構造の基本的な形式の把握（タスク、周期ハンドラなど）

- 走行体の組み立て、動作確認に手間がかかった
- だが、組み立てさせないと、構造や配線に関心を払わない

2.1.1 対面で使用した走行体

2つのモーターの駆動ユニット、バンパー、ライン監視部、側壁監視部、荷台（座席）、ホルダーアームを持つ。

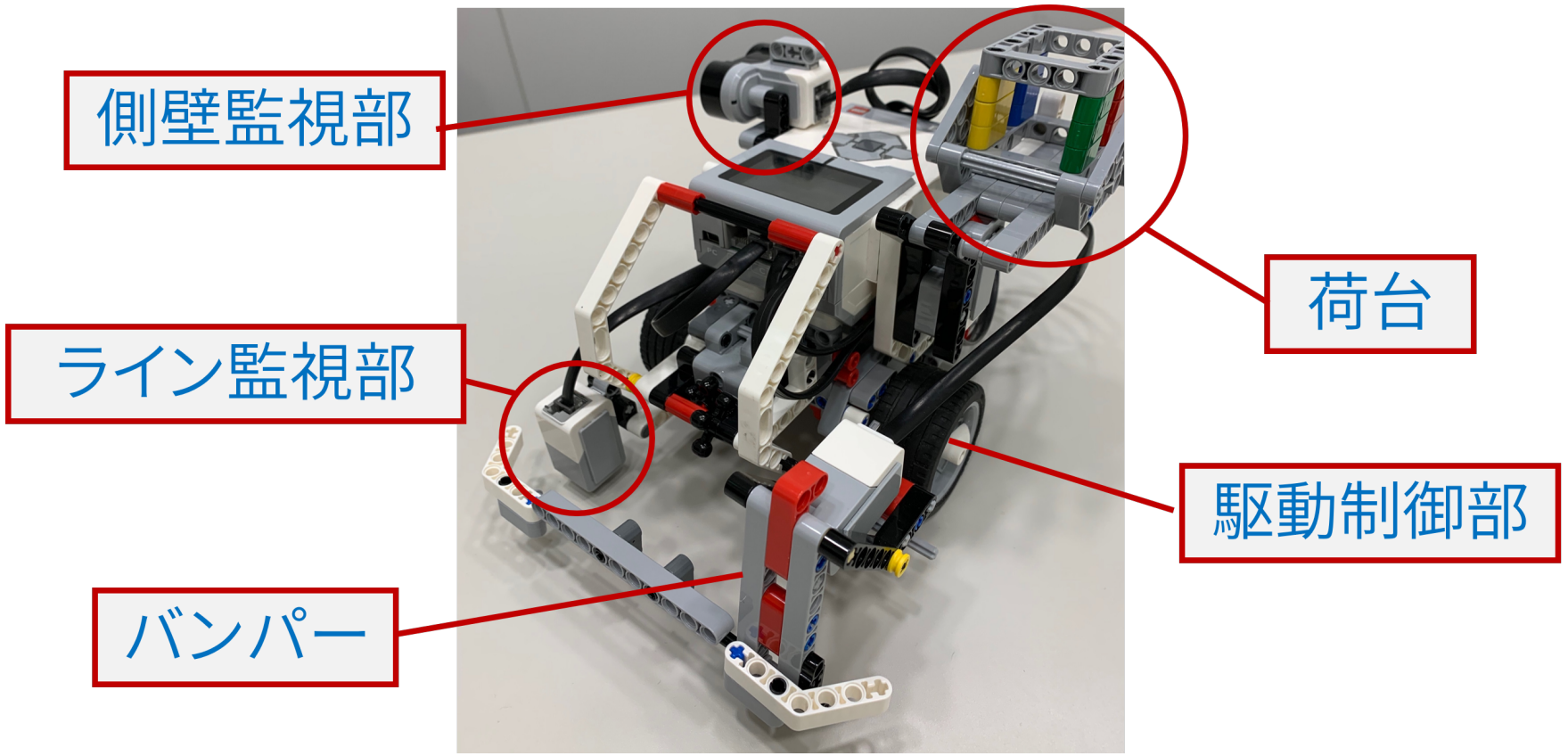


図 2.1 実機（EV3）の走行体の例

2.1.2 対面の演習の様子

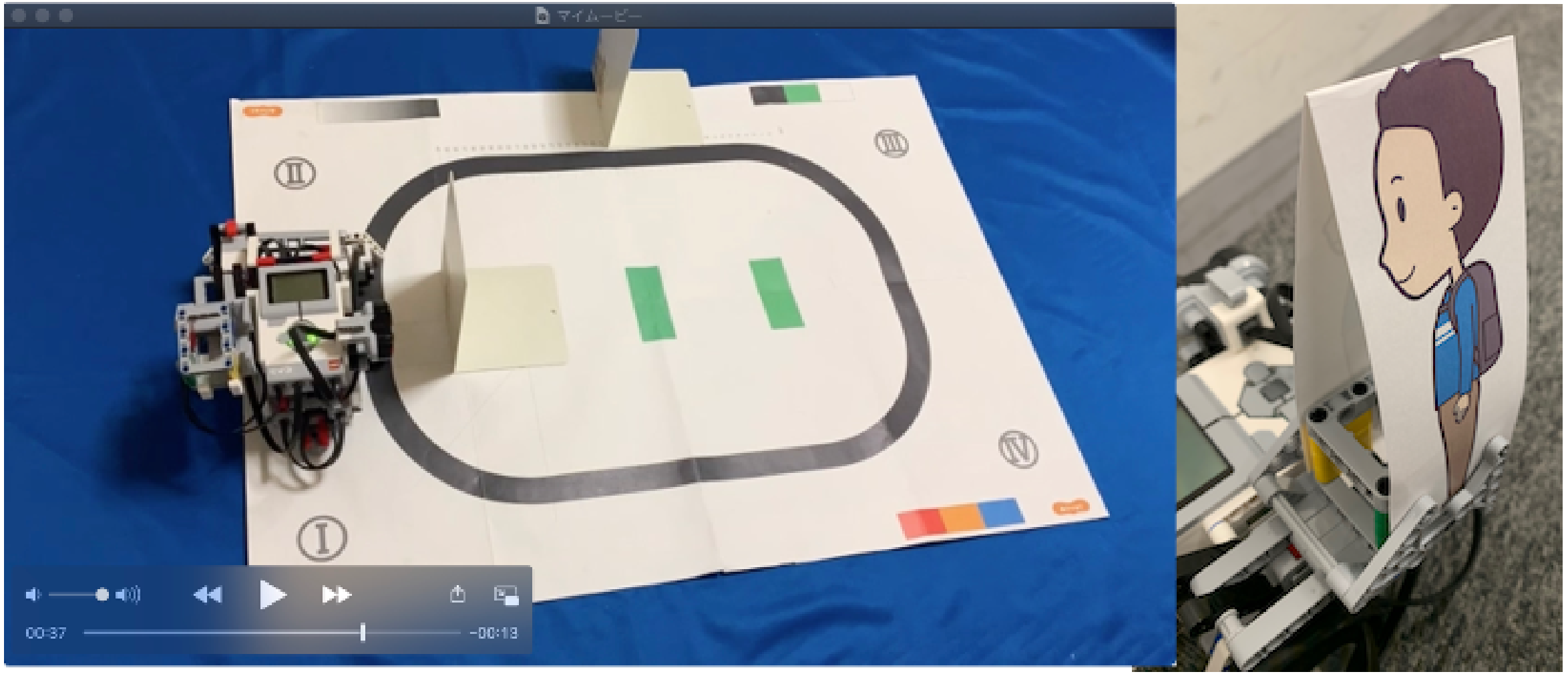


図 2.2 荷物を運搬（場合によっては人を輸送）する様子

図 2.3 経路から障害物を取り除きながら走行する様子

2.2 機器を動かす演習（オンライン）

- 各自のPCでシミュレーション環境「箱庭」を構築して演習
  - 環境構築でつまづいてしまい、履修を諦めた学生が少なくなかった
  - 現在は構築の大変さは軽減されつつある（が、学生のPCではシミュレーションとZoom同時はつらいようだ）
- **EV3のサンプル・プログラムが実行できる**
  - OSのAPI、センサーやモーターの使い方は実機と同じ

- × 環境構築ができない学部生が大変多い（WSLの導入でアウト）
- ◎ 一度環境が構築できれば、自宅でも演習できる
- △ 受講生が走行体やコースを変えるのは難しい（予め教材側で用意が必要）
- △ できあがったものを動かすので、構造や配線に関心が及ばない

2.2.1 箱庭[hakoniwa]の単体ロボットシミュレータを活用

箱庭は、CPU命令セットシミュレータをコアとする、IoTシステムを開発／提供するシミュレーション環境。担当する講義や研修の多くはMindstorms EV3を使っているので、主にEV3RTが使える「単体ロボットシミュレータ」環境を使っている。

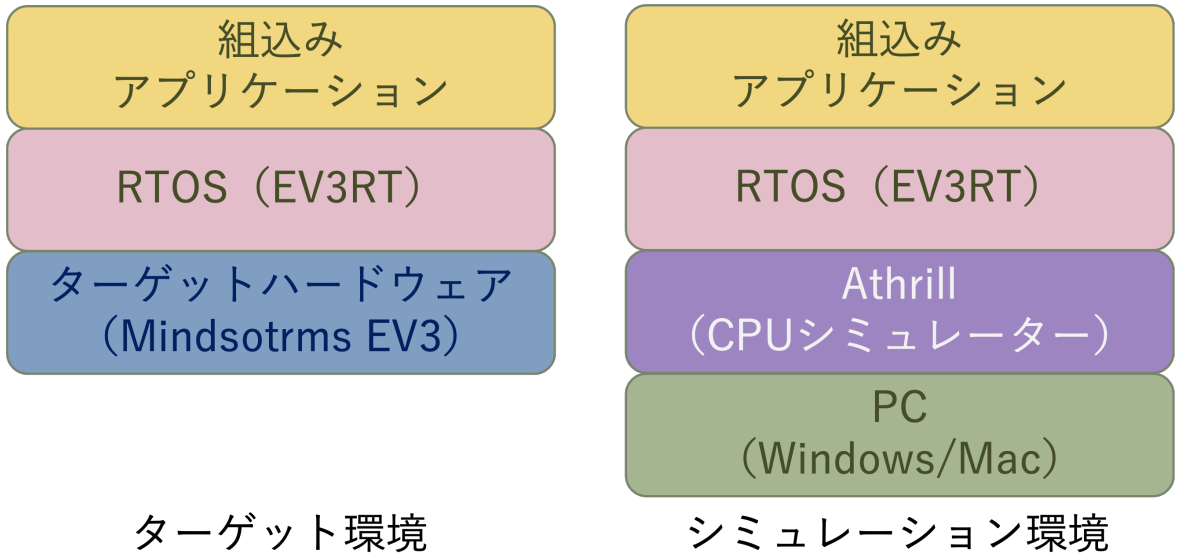


図 2.4 PC上でEV3の実機同じプログラムを実行できる

2.2.2 オンラインで使用した走行体

実機とほぼ同様の構成。

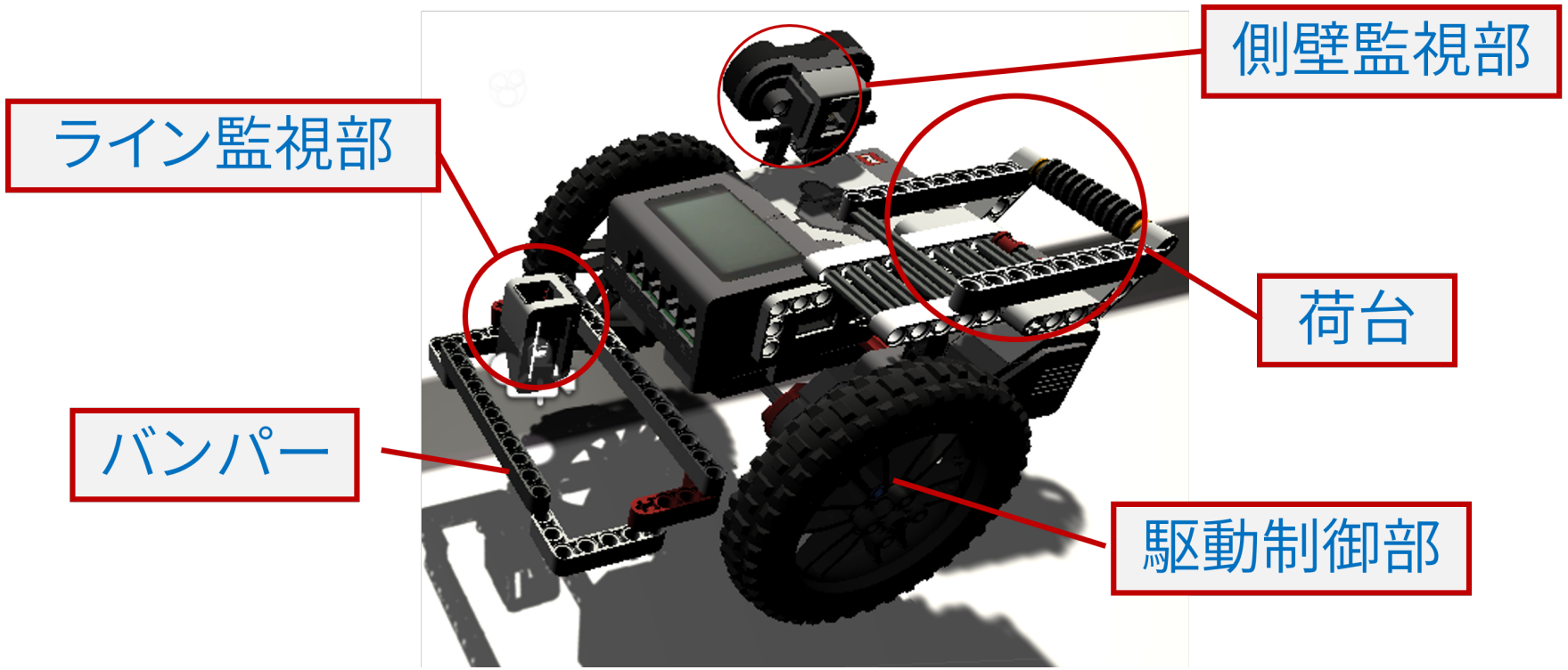


図 2.5 箱庭上の走行体の例

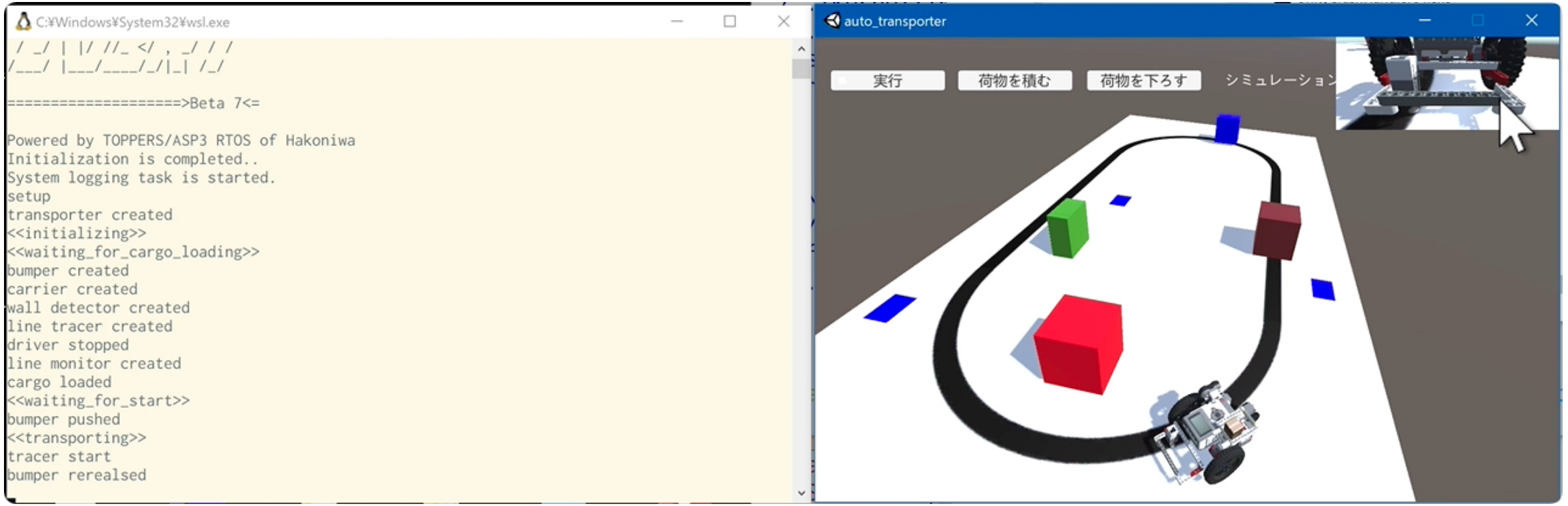


図 2.6 単体ロボットシミュレータを使って演習している様子

2.3 モデルを作成する演習（対面）

使用するもの

- 模造紙＋ポストイット、グループで作成（ 図 2.7 、図 2.8 ）
- astah* や BridgePoint で作成、グループで作成
  - 個人でやってもらう場合や、すでにツールを利用している場合
    - 使い慣れていないときは、ツールは使わない（操作の演習はモデリングの演習にならない）

進め方

- 最初は講師が取っ掛かりをつけて、真似してもらう
- まだできていない（or さっぱり??な）状況なら、すぐ全体に対して共有
  - astah* を使っているなら、プロジェクターに表示して共有
  - できていない受講生は、講師と一緒にモデルを作れるのでおトク
- ほかのグループ（または人）は、それを見ながら自分たちのモデルを見直す

対面で実施している様子

https://ja.astahblog.com/2017/05/23/uml-seminar-report/

図 2.7 模造紙とポストイットでモデルを作成している様子

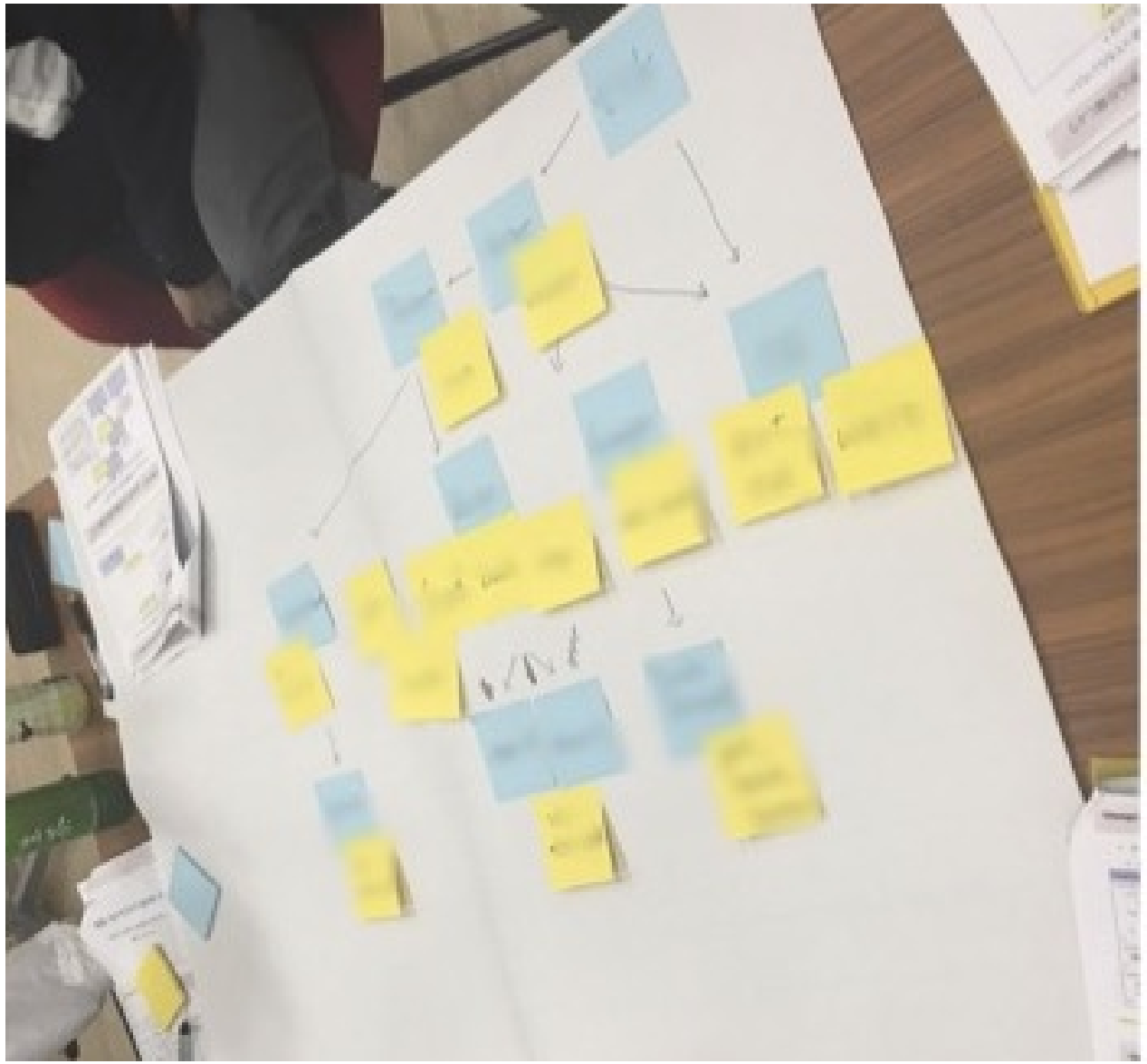


図 2.8 模造紙とポストイットで作成したモデル図

2.4 モデルを作成する演習（オンライン）

グループでやるのか？

- Zoomのブレイクアウトルームでグループワーク
- 個人のワークに変更する場合もある

モデルはどうやって作るのか？

- Miro でモデルを共有して作成、グループで作成
  - だれかのモデルを画面共有して議論
- astah* でモデルを作成
  - だれかのモデルを画面共有して議論
- PowerPointの作図機能でモデルを作成
  - だれかのモデルを画面共有して議論
- 上記のいずれの場合も、個人でモデルを作成する方法に変えられる

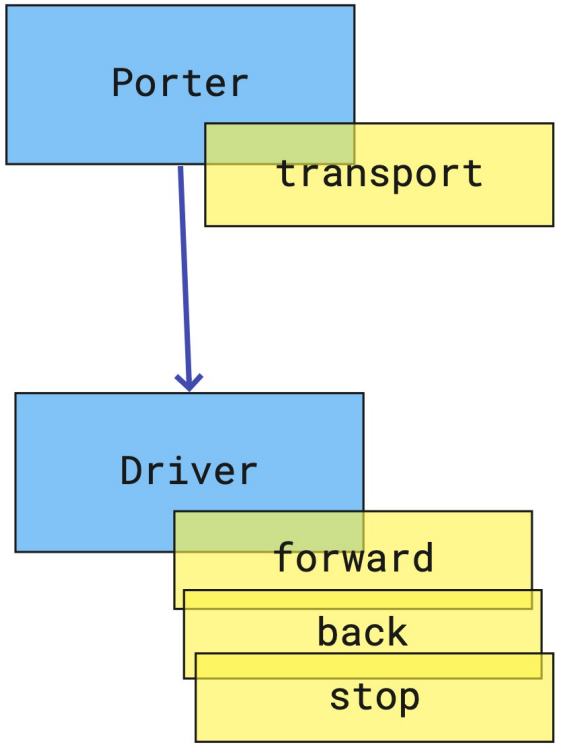
2.4.1 Miroをつかうとき

- Miroのワークスペースは、グループで分けず、全てのグループで1つ
  - 広い1つのワークスペースをフレームなど使って分けて使う
- あらかじめ、図の描き方とサンプルを用意しておく（ 図 2.9 ）
- グループにブレイクアウトルームを割り当てる
- グループに分かれてワークする（ 図 2.10 ）
  - 敵情視察は自由

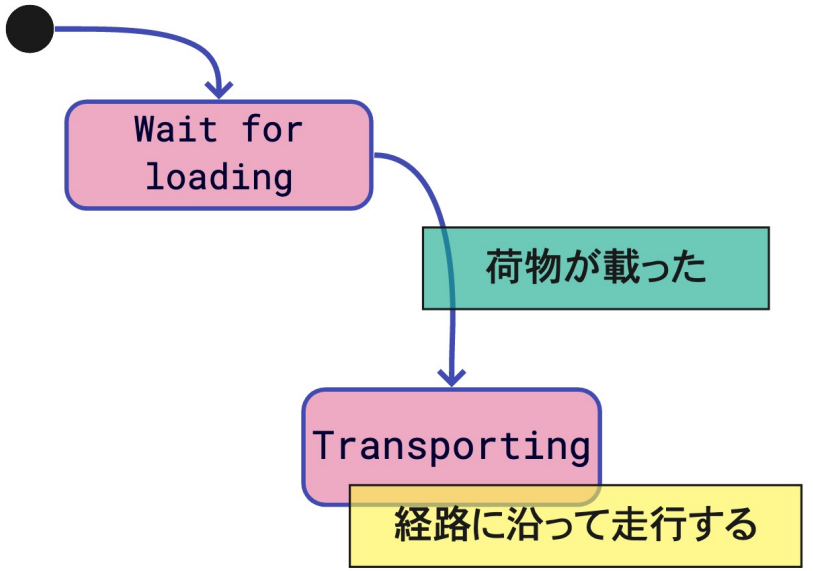
> Miroの操作に慣れていない人がいれば、操作の練習も必要

オンラインホワイトボード「Miro」

https://miro.com/



これがクラス図を演習で作るときのイメージ



これがステートマシン図を演習で作るときのイメージ

miro

図 2.9 Miroで図を描くときの記法ガイド

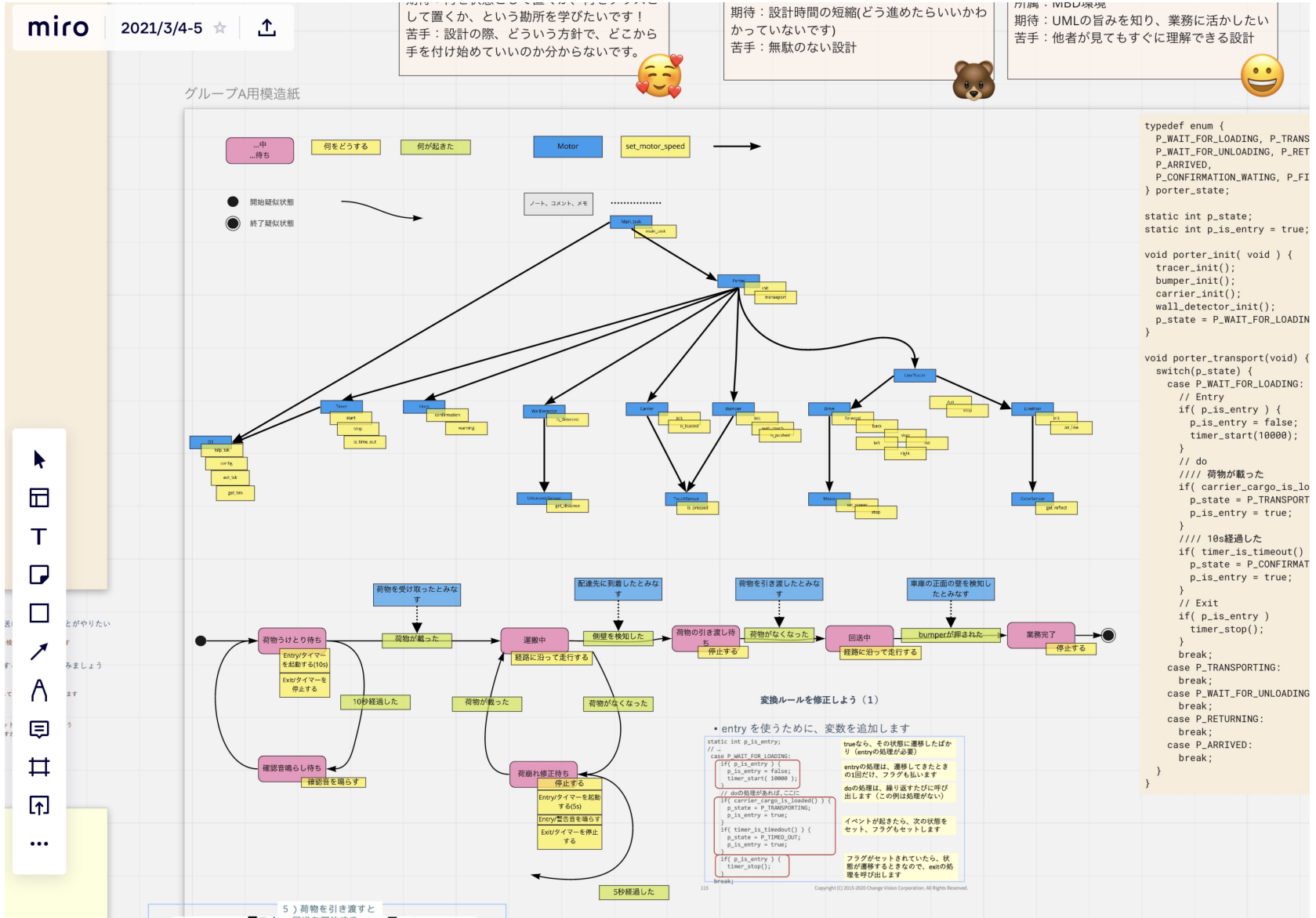


図 2.10 Miroで演習している様子（部分）

2.4.2 Astah*をつかうとき

- グループに分かれるとき
  - グループにブレイクアウトルームを割り当てる
  - だれかがモデルを表示して、グループで画面共有
  - 講師がブレイクアウトルームを見て回る
- 個人の演習のとき
  - 最初にだれかに画面共有してもらう
  - 共有した人に講師が指示をだしながら、プロジェクト作成、サンプルのインポートなどは全員で進める
  - モデルを作る最初のとりかかりは、共有した人と講師とで問答しながら作成
  - 軌道にのってきたら、各自で演習
    - 希望があれば最初と同じようにして共有して講師と一緒に

2.4.3 PowerPointをつかうとき

基本的に、模造紙を使うときと同じ。

- ワークブックになる `pptx` ファイルを配布
  - 各自、あるいはグループでそのファイルを編集する
- ワークブックには、図の描き方のルールを記載しておく（ 図 2.11 、図 2.13 ）
- 作成した例（ 図 2.12 、図 2.14 ）

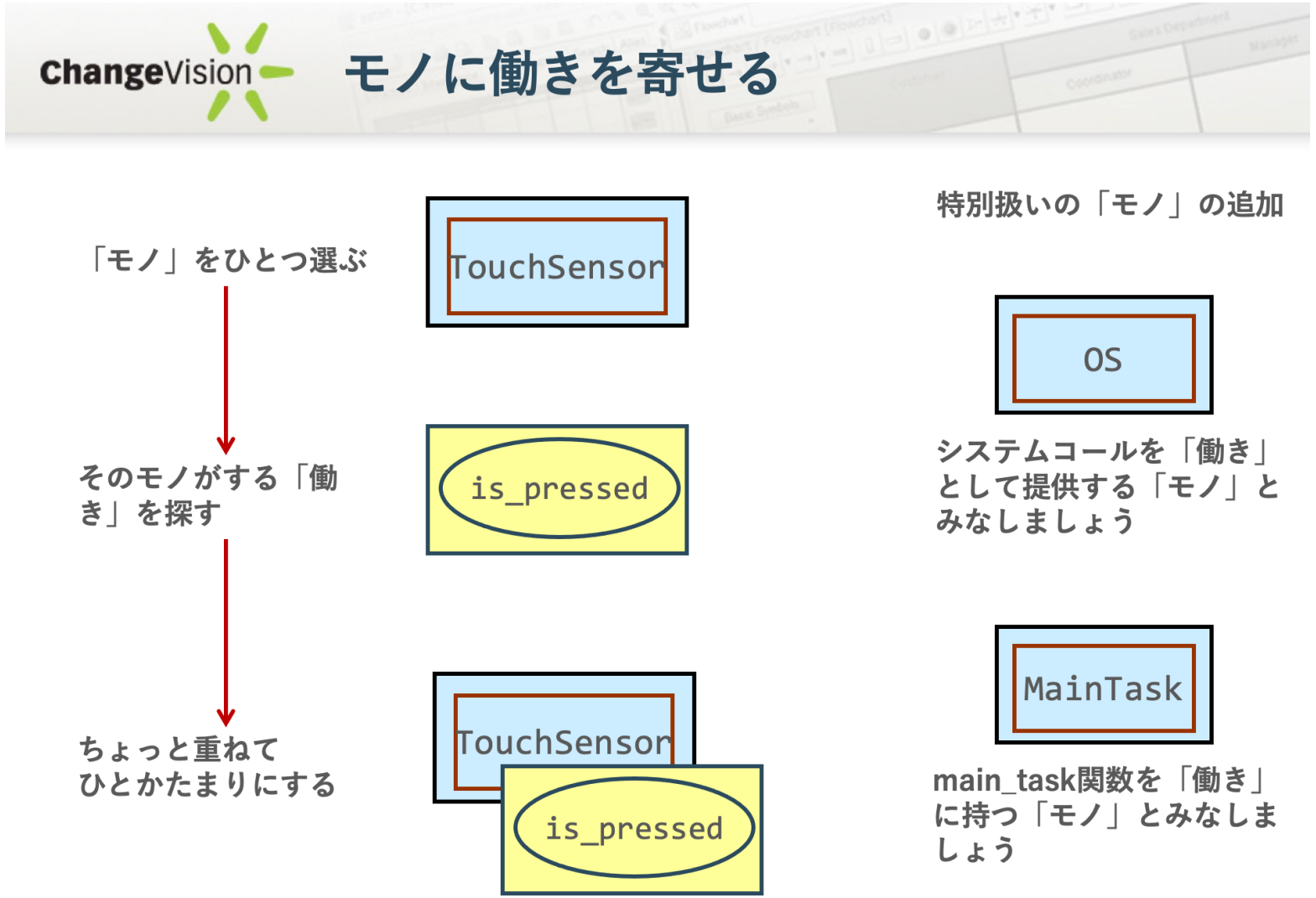


図 2.11 ワークブック中のクラス図の描き方のルール（一部）

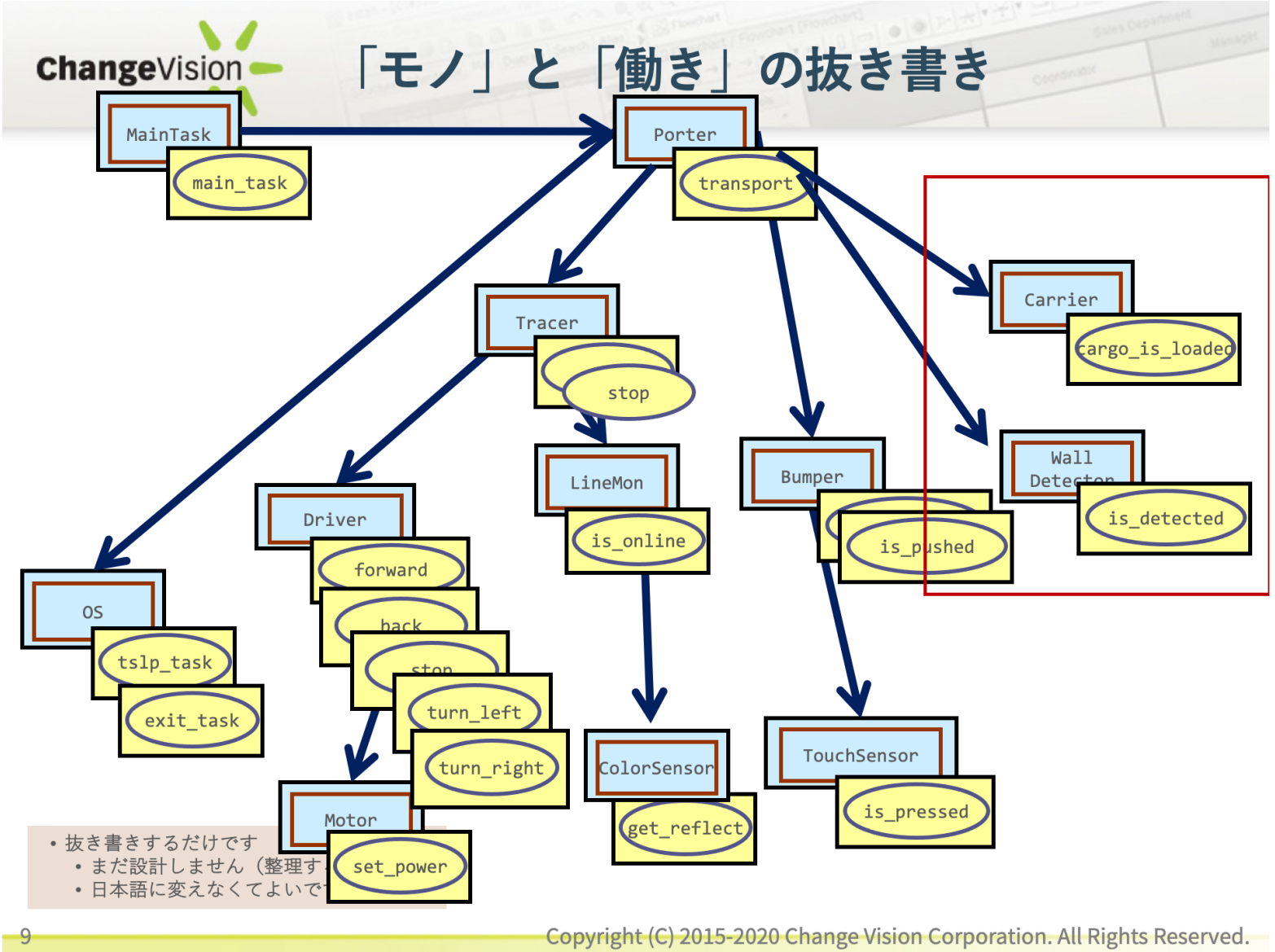


図 2.12 演習でワークブック上にクラス図を作成している様子

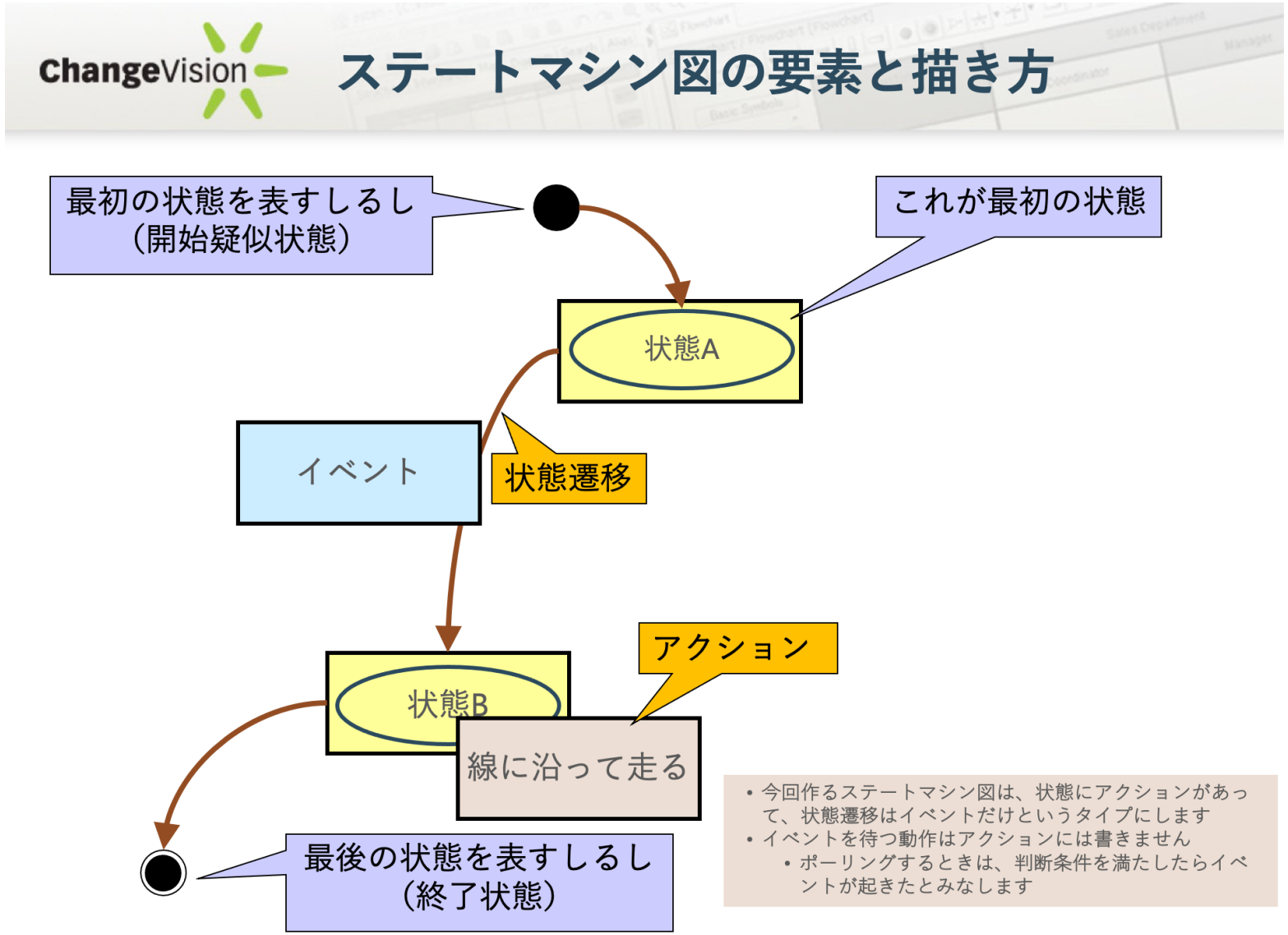


図 2.13 ワークブック中のステートマシン図の描き方のルール

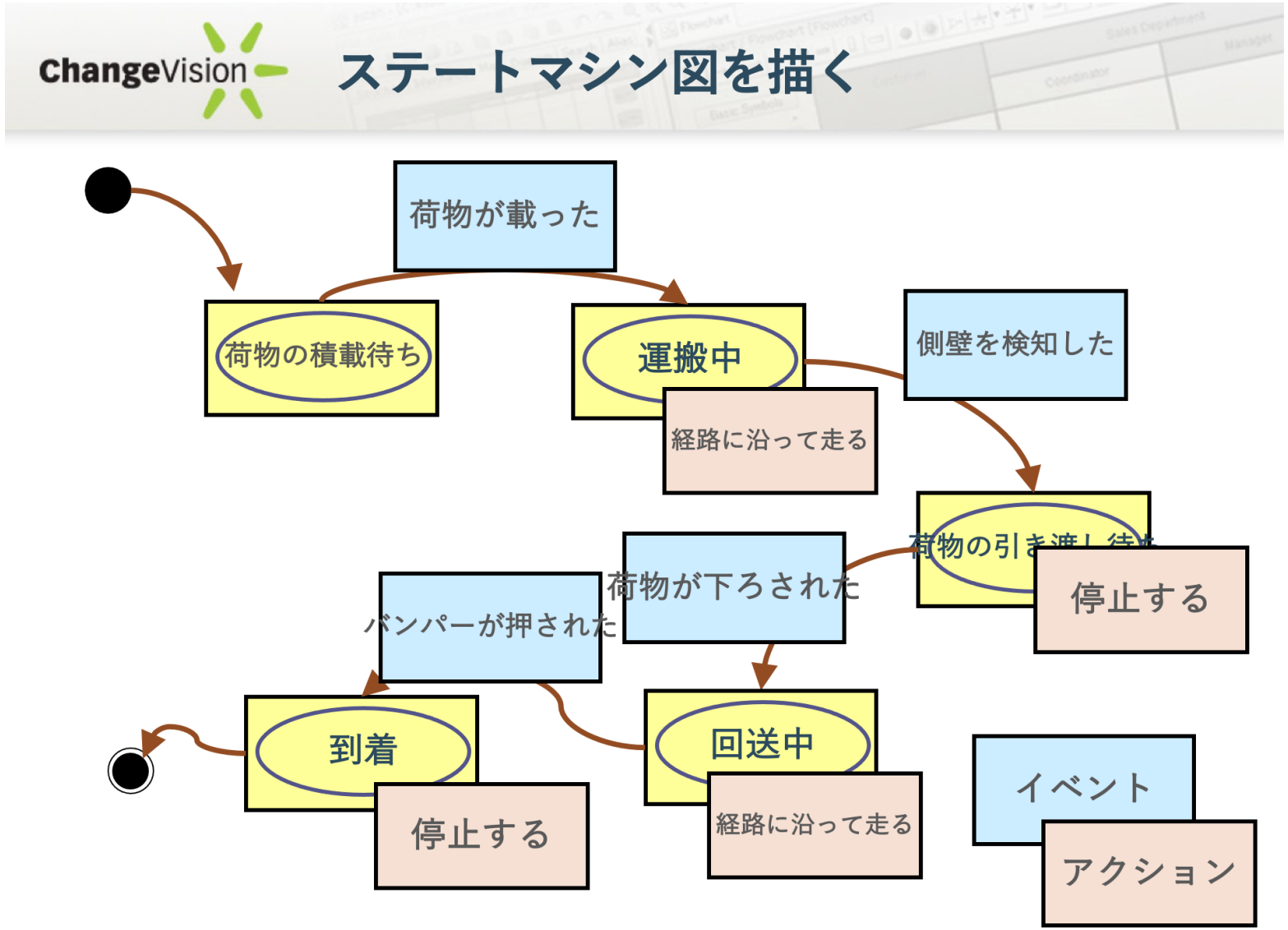


図 2.14 演習でワークブック上にステートマシン図を作成している様子

2.5 モデルからコードを作成する演習

- 基本的に、対面とオンラインで変わらない
- **「コードの書き間違いで動かない」問題と「設計の間違い、検討不足」を混同させない**
  - 「コードの書き間違い」は変換ルールの利用ミス、「設計の間違い、検討不足」はモデルのミスになるよう実装方法を制約する

2.5.1 対面で実施したとき

制約のある方法でコードを書く

- クラス図とステートマシン図からコードを作成
  - 簡便なルールでよいので、みんなが決まった方法でコードに変換する（ 図 2.15 ）
- 自分たちの課題のモデルを作る前に、かんたんなサンプルで、みんなでいっしょに手動変換を試しておく
  - このルールを使う前提で設計するように仕向ける

```
void auto_transporter_transport(void) {
  switch(p_state) {
  // 略
 case WAIT_FOR_UNLOADING:
    line_tracer_stop();
    if (! carrier_cargo_is_loaded()) {
      p_state = RETURNING;
    }
    break;
  case RETURNING:
    line_tracer_run();
    if (bumper_is_pushed()) {
      p_state = ARRIVED;
    }
    break;
  case ARRIVED:
    line_tracer_stop();
    break;
  }
```

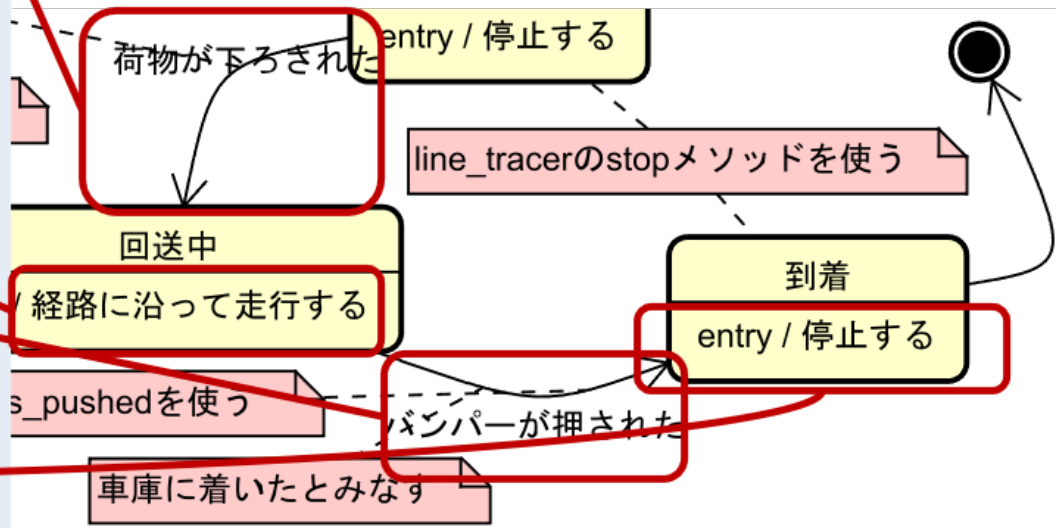


図 2.15 簡便なルールで図の要素とコードを対応づけておく

グループでも個人でも同じ方法で

- モデルをみながらグループで作成
  - 一つのソースをみなで編集
- 個人で作成する場合
  - 手が止まっている人のエディタ画面を共有して講師と一緒に編集

コード生成を使って実施する場合

- BridgePointでコード生成
  - 手作業の演習と同様のモデル作成とコード生成を実施
  - 生成したコードで NXTやEV3 を動かす（トップエスイー）（ 図 2.16 ）
  - 生成したコードで Raspberry Pi を動かす（スマートエスイー）

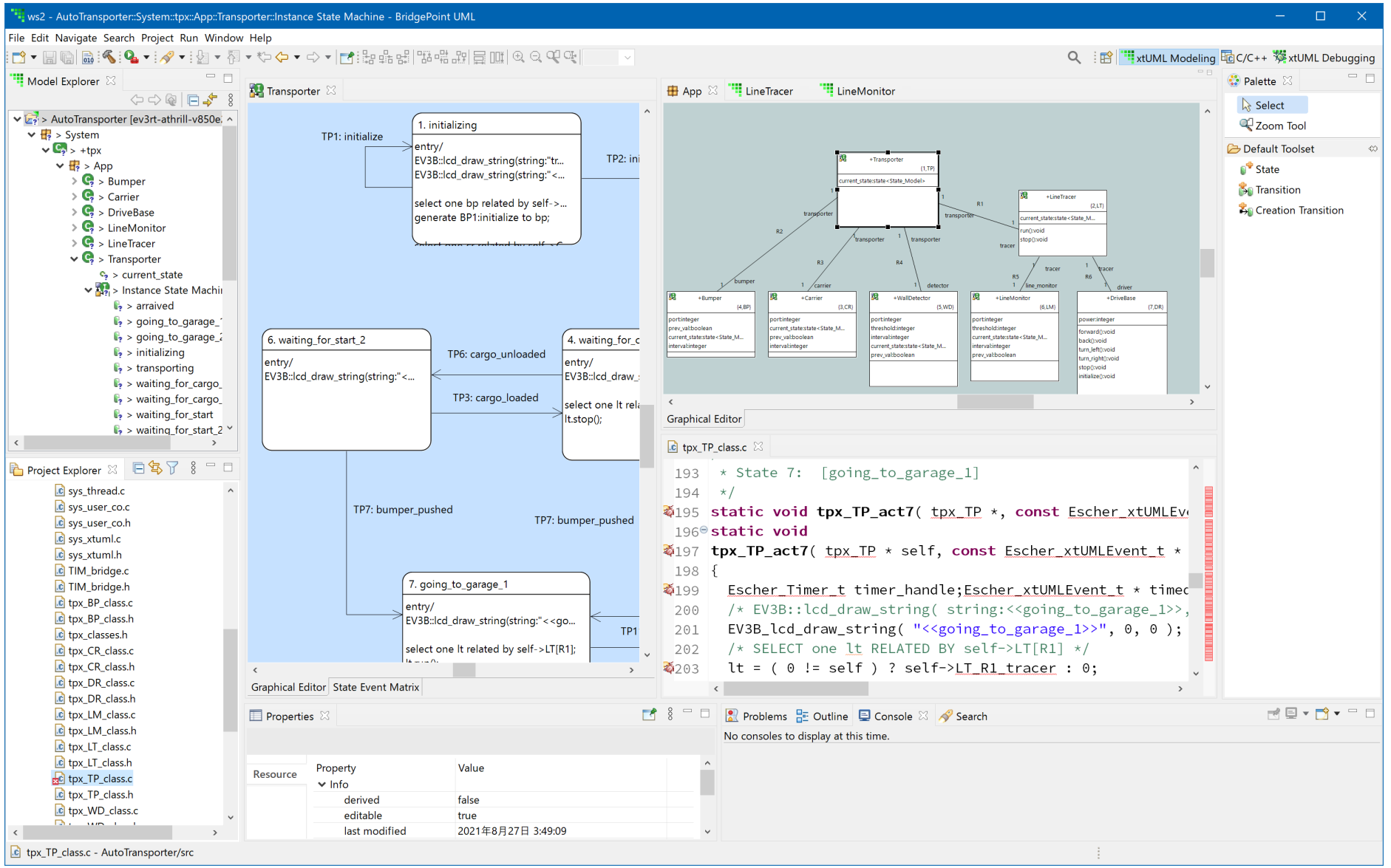


図 2.16 BridgePointでモデルを作成しコードを生成（トップエスイー）

2.5.2 コードのレビューは、ほぼモデルのレビューになる

- まず、ルール通り（返還による実装か）を確認する
  - 描いた図のとおりなら、どんなコードになるか
- モデル（たいていはステートマシン図）で表している内容の不備や検討不足を検討する
  - 書きたいと思っているコードのように動作させたいなら、どんなモデルを描くべきか議論する
- モデルを直し、それに合わせてコードを直して再実行

2.5.3 オンラインで実施したとき

- ルールを使ってコードを作成する方法を使うのは、実機を使う場合と同じ
- オンラインの場合にも、コードとモデルを画面共有し、講師と一緒にコードを書く・直す
  - 周りもそれを見ながら自分のコードを見直す
- 動作確認は、各自の箱庭の環境で実施する
  - 成果発表でのデモも、グループの誰かの箱庭を使う

対面で実機がある場合（ハイブリット形式）の場合

- リモートの受講生がビルドしたローダブルモジュールを、会場（教室）にいる受講生へ送って実行してもらう
  - リモートの学生に動作の様子を送るには、外付けのUSBカメラがあるとよい
- ファイル共有は、GoogleDeive等、各校のインフラが提供する方法を使う

3. モデル・コード・レポートの対応

オンラインで、たくさんのモデルやコードをみるには。

3.1 演習成果の評価はめんどう

演習では、モデル図やコードなどの成果物とデモの結果を見聞する必要がある。

- モデル図やコードをレポートとして出してもらうのが大変
  - `docx` へのはりつけも、レポートの構成、体裁もバラバラでひどい
- たくさん（100名）のレポートを紙や `docx` でもらうと、見る方も大変
  - とくに、フィードバックをするのが別手段になり、レポートとフィードバックの対応関係を自分で掌握するのが大変

- なにか、オンラインで成果を確認し、フィードバックしやすいしくみが必要

3.2 GitHub Classroomの活用

ソフトウェア開発に関するコースを管理したり、コースに参加したりするのに、GitHubが使える。

GitHub Classroom

https://classroom.github.com/classrooms

GitHub Classroomを使用したコースワークの管理

https://docs.github.com/ja/education/manage-coursework-with-github-classroom

- クラスに配布する教材をリポジトリに作成して、配布できる
- 各自は、配布したリポジトリを使って演習してプッシュする
- プッシュした結果を講師が確認して、評価やコメントを返す
- Google Classroom や Moodleとも連携できる（らしい）

3.3 GitHub Classroomから教材を受け取る

教材のURLにアクセスし、「Authorize GitHub Classroom」をクリックする（ 図 3.1 ）。

演習用教材のURL

https://classroom.github.com/a/gq_r2KQY

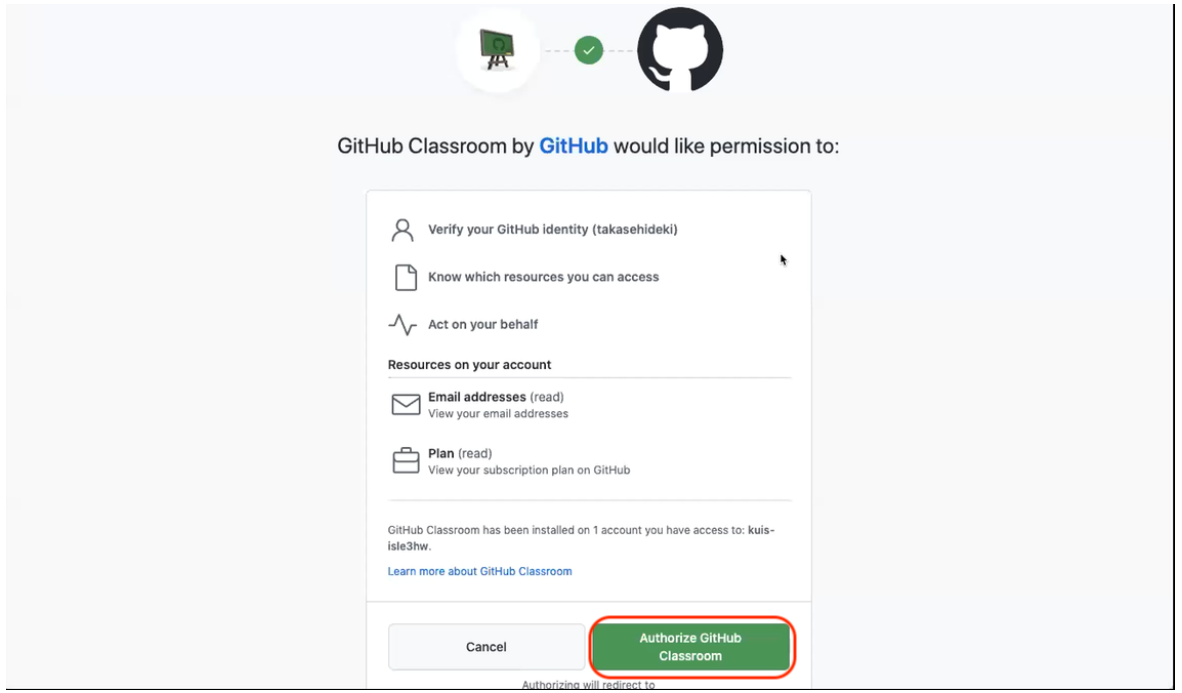


図 3.1 GitHub Classroomを承認する

認証を経て教材の受け入れ画面になる。「Accept this assignment」をクリックする(図 3.2)。

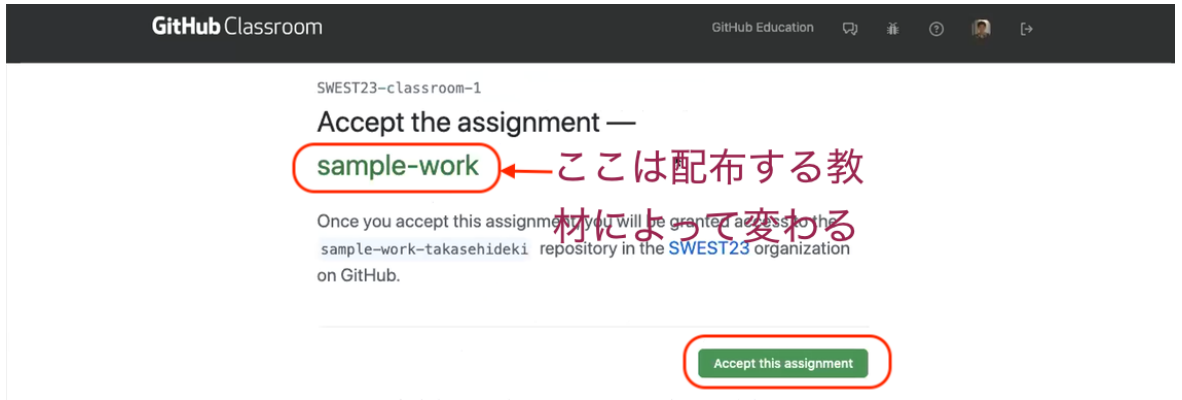


図 3.2 教材を受け取る

リポジトリをフォークするので、少し待ってページを再読み込みする(図 3.3)。

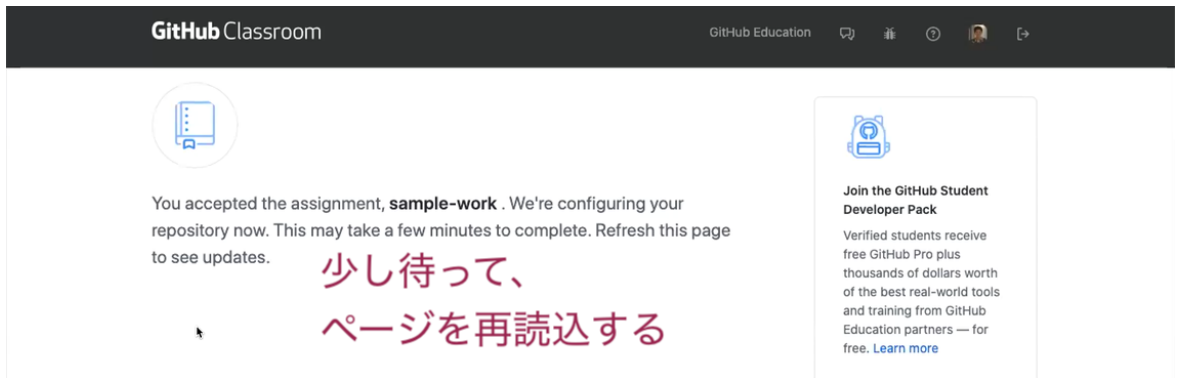


図 3.3 リポジトリの準備するまで少し待つ

すると、準備されたリポジトリのURLが表示されるので、クリックして開く(図 3.4)。

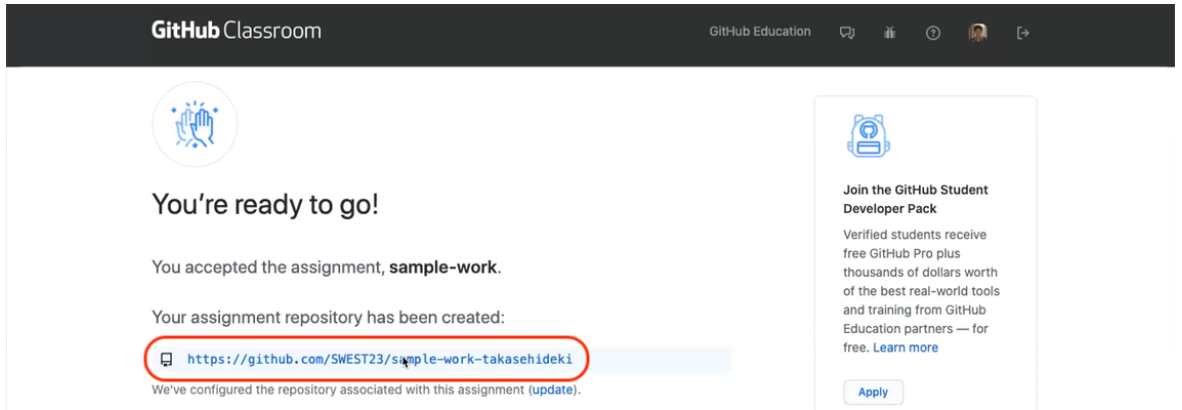


図 3.4 教材のリポジトリが準備できた

配付した教材の個人用リポジトリが作成される(図 3.5)。

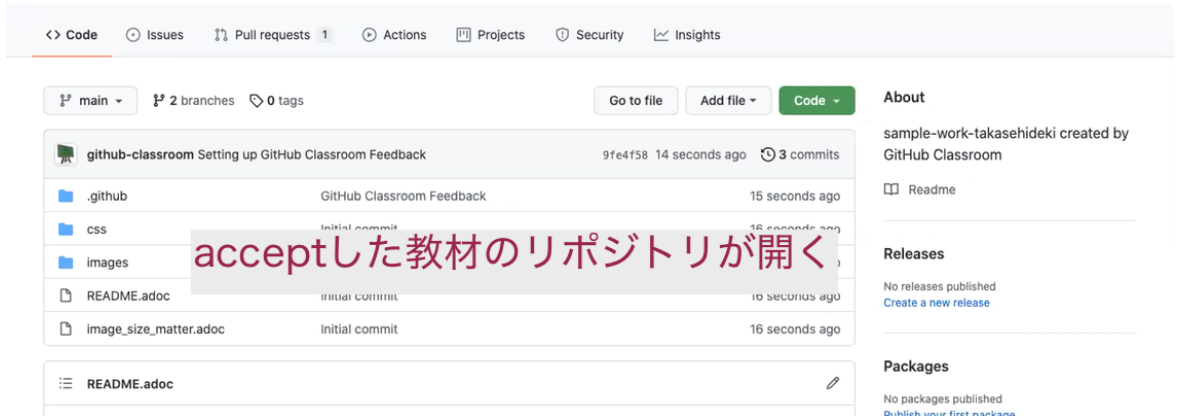


図 3.5 教材のリポジトリが準備できた

別途、GitHubのアカウントに使ったメールアドレス宛に、リポジトリへの招待を確認するメールとフィードバック用に使うプルリクエストについての注意を促すメールが届く(表 3.1)。

表 3.1 GitHub Classroomから届くメールの例

GitHub

@github-classroom[bot] has invited you to collaborate on the SWEST23/sample-work-takasehideki repository

You can accept or decline this invitation. You can also visit @github-classroom[bot] to learn a bit more about them.

This invitation will expire in 7 days.

View invitation

Note: This invitation was intended for **takasehideki@hal.ipc.i.u-tokyo.ac.jp**. If you were not expecting this invitation, you can ignore this email. If @github-classroom[bot] is sending you too many emails, you can block them or report abuse.

Getting a 404 error? Make sure you're signed in as **takasehideki**.

Button not working? Copy and paste this link into your browser:
https://github.com/SWEST23/sample-work-takasehideki/invitations

Manage your GitHub email preferences
Terms • Privacy • Sign in to GitHub

👋! GitHub Classroom created this pull request as a place for your teacher to leave feedback on your work. It will update automatically. **Don't close or merge this pull request**, unless you're instructed to do so by your teacher.

In this pull request, your teacher can leave comments and feedback on your code. Click the **Subscribe** button to be notified if that happens.

Click the **Files changed** or **Commits** tab to see all of the changes pushed to `main` since the assignment started. Your teacher can see this too.

▸ **Notes for teachers**

Subscribed: @takasehideki

You can view, comment on, or merge this pull request online at:

https://github.com/SWEST23/sample-work-takasehideki/pull/1

Commit Summary

- Setting up GitHub Classroom Feedback

File Changes

Patch Links:

- https://github.com/SWEST23/sample-work-takasehideki/pull/1.patch
- https://github.com/SWEST23/sample-work-takasehideki/pull/1.diff

—
You are receiving this because you were mentioned.
Reply to this email directly, view it on GitHub, or unsubscribe.

3.4 GitHub Classroomを使った演習

演習に使う教材

modeling_with_pptx

ここからは、配付した教材を使って演習する(教材を使った演習の進め方)。

教材を使った演習の進め方

1. GitHubからローカルにCloneする
2. Cloneしたリポジトリの `README.adoc` を Asciidoctor Browser Extension を追加したブラウザで開く
3. `README.adoc` をテキストエディタで編集しながら、演習を進める

4. オンライン演習の良し悪し

4.1 実は、学生の反応は厳しいものだった…

演習環境が構築できない、リモートでモデルの演習はつらい、コードを書いて動かすところをグループでやるのは難しいといった指摘をたくさん受けた。

4.1.1 厳しいフィードバックの例

- 実機ほどには「動いた！感」がないので盛り上がらない
  - 動かす楽しさで盛り上がって、多少の大変さを克服してもらうねらいが…
- PCの性能が低いと、動かすのが億劫になる
  - 学生のPCだと、Zoomで画面共有すると、さらに重くなる（デモが動かない）
- 環境構築ができない
  - 演習を始めるまでに至らず（そして履修をあきらめてしまう）
  - WSLやDockerが使えるようにならない（OSが古い、メモリが足りない、難しいなど）
- 動かないときの原因がわからない
  - 箱庭の環境設定ファイルの誤りなどがあっても、原因がどこにあるのかわからない

4.1.2 実機を使う「体験」の代わりにはならない

- 実機のときのように夢中にはなってくれない
  - 学生のPCでは、処理が重いので動かすのが億劫
  - シミュレーションの環境が起動、動作とも重いようだ
- 1度動かすと、あとはあまり動かす気持ちにならない
  - 実機ほどは外乱の影響がないので、動きに変化がない
  - 他の人の動かしている様子が、横目に入ってこないので、焦りがない
  - 他の人のできが見えないので、自分の間違いや不出来に気づかない

- × 実機で演習できないから、代わりに使うという考えは、うまくいかない
- × 組み立てていないので、構造を把握できていない
- △ 好きに組み立てて試す楽しさはいまひとつ（Unityのモデルを自分で作れない）

4.1.3 実機では軽んじられる側面は、試しやすい

- 設計やコード作成の演習としては便利
  - 箱庭は、どこでも動作検証に使えるのがよい
- 実機では感じない・観測できないことを試せる
  - 実機では外乱やバッテリー切れは、都合よく起こせないが、シミュレーションなら制御できる
  - ユニットに仕込む、シミュレーションのティックを変えるなど
- 実機では環境のせいにするが、そのいいわけが使えない
  - 周囲の明るさが違うせい…、コースにシワがあったから…など
  - そのような変化がない環境で動作できるので、別の原因だと気づける

- ◎ 個人で自宅で演習するには向いている（実機は全員には配れない）
- ◎ 作為的に外乱や変動を起こせる
- △ 周囲の状況の変化や外乱があることを意識しずらい

まとめ

オンラインでも基本は対面と同じ

- 基本的には、オンラインでのモデリングでも、対面と変わらない
  - モデル図の作成方法や共有方法に工夫が必要
  - 「動かす体験」には向かない場合がある
- コードを書いて動かすには箱庭のような実機の代替があるとよい
  - 各自が自宅でも試せるのは、演習室に来たときだけ動かせるより有利なところもある
- 実装は変換ルールによる制約をつけてやってもらう
  - ルールを使うことで、コーディングの問題と設計の問題を分離しやすくなる
  - 実装に自信がなくても、サンプルとルールの真似をすればまぁまぁいける

最初のハードルは環境構築

- 環境構築に不慣れな学生には箱庭の環境を用意すること自体が難しい
- 学生のPCを使う場合に、資源が不足する場合がある
- このようなことが理由で履修を諦めてしまう学生も多い

モデルやコードの驍勇やレポートの対応策

- モデルやコードをのレポートには、GitHub Classroomを使う
  - 課題の配布、回収、フィードバックがGitHubでできる
  - 学部の学生にも、多少はGitを使ってもらえるようになる
- レポートは、配布リポジトリのファイルを書き換えてもらって済ませる
  - asciidoctorやmarkdownで書いてもらい、`docx` のお世話にならない

参考文献

- [classroom] GitHub. GitHub Classroom.
  - https://docs.github.com/ja/education/manage-coursework-with-github-classroom
- [miro]
  - https://miro.com/
- [hakoniwa] TOPPERSTOPPERSプロジェクト箱庭WG. 箱庭 IoT／自動運転時代の仮想シミュレーション環境.
  - https://toppers.github.io/hakoniwa/docs/
- [UMLSPEC] OMG 統一モデリング言語（UNIFIED MODELING LANGUAGE）.
  - https://www.omg.org/spec/UML/
- [ダイアグラム別UML徹底活用 第2版] 井上. ダイアグラム別UML徹底活用 第2版.
  - https://www.shoeisha.co.jp/book/detail/9784798118444
- [ev3rtcapi] TOPPERS. EV3RT C API Reference.
  - http://www.toppers.jp/ev3pf/EV3RT_C_API_Reference/index.html

- [ev3rt04] 李. EV3RTの内部構造と進んだ使い方.
  - https://www.toppers.jp/docs/etrobo15/ev3rt_seminar_04.pdf
- [ev3rt01] 石川. EV3RTの概要.
  - https://www.toppers.jp/docs/etrobo15/ev3rt_seminar_01.pdf

注意事項

諸注意

- 本文書は、久保秋真（作成者）が編集したもので、本文書に関する権利、責任は作成者が保有します。
- 本文書に記載されている情報は、2021年7月現在のものです。URLやWebページの内容、各種の情報は、利用時には変更されている可能性があります。
- 本文書は演習用テキストとしての使用を想定し、内容等は予告なしに変更することがあります。また、作成者がその内容を保証するものではありません。
- 本文書の内容に誤りや不正確な記述がある場合も、作成者は一切の責任を負いません。
- 本文書に記載の内容や実施結果からいかなる損害が生じても、作成者は責任を負いかねますので、あらかじめご了承ください。
- 本文書の複製、保存については、作成者の同意を得てください。

商標等について

- Windows 並びに同社の各製品名は米国マイクロソフトコーポレーションの米国およびその他の国における登録商標です。
- Apple、iCloud、iPad、iPhone、Mac、Macintosh、macOSは、米国およびその他の国々で登録されたApple Inc.の商標です。
- Linuxは Linus Torvalds氏、米国及びその他の国における登録商標 あるいは商標です。
- UNIXは、X/Open Company Limitedが独占的にライセンスしている米国ならびに他の国における登録商標です。
- LEGO、LEGOロゴ、MINDSTORMS、MINDSTORMSロゴは、LEGOグループの商標または著作権です。
- その他、本書に記載されている社名、製品名、ブランド名、システム名などは、一般に商標または登録商標でそれぞれ帰属者の所有物です。
- 本文中では © 、® 、™ 、は表示していません。

オンラインでの組込み教育・モデリング教育について
講演資料

発行日：2021-09-03
バージョン：pdf_0115
作成者：久保秋 真
本書の内容に関する質問等がありましたら、作成者までお知らせください。