

ETロボコン2022

デベロッパー部門 アドバンストクラス

チームNo. 169 チーム名: ISC走る.exe

地区: 南関東

地域: 神奈川県横浜市



所属: 学校法人 岩崎学園 情報科学専門学校

チーム紹介、目標、意気込み

ISC走る.exeは学校法人 岩崎学園 情報科学専門学校の3年生と4年生の学生で構成されたチームです。

私たちは学校の授業の一環でETロボコンを取り組んでいますが、対面での活動が難しかったり、授業だけだと時間が足りないので、オンラインで集まって作業をしています。

ほとんどのチームメンバーが初めてのETロボコンなので手探りで頑張っています。

ISC走る.exeの目標は

「総合タイム0秒」

目標達成を目指してチーム一丸頑張ります！

モデルの概要

1. 開発工数の削減

ETロボコン経験者及びプログラミング言語を巧みに扱える生徒は少ないためできるだけ定義や設計を簡素化し開発工数を抑える必要があるため設計モデルにシングルトンクラスを設ける等を行いました。

2. 平行処理

PC側と走行体側のリソースを両方使い、通信で情報をやり取りする。
そうすることで、走行の安定化と経路探索の高速化を図った。

モデルの構成

1. 要求モデル

ETロボコン2022の審査規約と競技規約を踏まえて、このチームの全員で話し合い実現可能な目標を設定した。目標達成のために必要な機能をユースケース図に示した。目標とユースケース分析で各機能ごとに必要な要件を検討し要件定義を行った。

2. 分析モデル

ゲームエリア攻略のための走行体の動作定義を図と文章でまとめた。チームでゲームエリア攻略指針を決め、それを攻略していく解法を定めた。

解法を実現するために必要な要素を定義し、クラス図に示した。

3. 設計モデル

各種要求モデル、分析モデルからパッケージの構造を模索し、パッケージ内の構造について示した。
走行体による区画制御の詳細な状態遷移を示した。
PCと走行体との通信を重点に置いてシーケンス図を示した。

4. 制御モデル

様々な観点から出た課題に対し対策を考案し、実施した。
人口知能を用いたブロックの座標及び色取得の課題を重点に記載した。

1. 要求モデル



1-1. 攻略方針

開発時の走行テスト結果や試走会の結果を基に図1-1で示すエリアの目標攻略時間を設定する。これらの攻略目標時間から目標総合タイムを決める。

競技規約をもとに、赤色の「走行エリア」と青色の「ゲームエリア」の2つのエリアに定義したものと示す。[図1-1]

「走行エリア」：スタートラインからゴールラインの範囲

「ゲームエリア」：ゴールラインから「ブロック de お片付け」を行うエリアの範囲

目標：総合タイム±0秒を獲得を目指して詳細な目標と方針を下記の表に示す

目標項目	目標秒数	攻略方針
走行エリアのタイム	10秒	可能な限り早くゴールライン到達を目指す。
ゲームエリアのタイム	110秒	ボーナスブロックを先に攻略し、残りのブロックを1個ずつ丁寧にブロックベースエリアに運ぶ。
ボーナスタイム	-10秒	10秒以上のボーナスタイムを獲得する。
総合タイム	±0秒	目標走行エリアのタイムを取り、可能な限りボーナスタイムを稼ぐ。

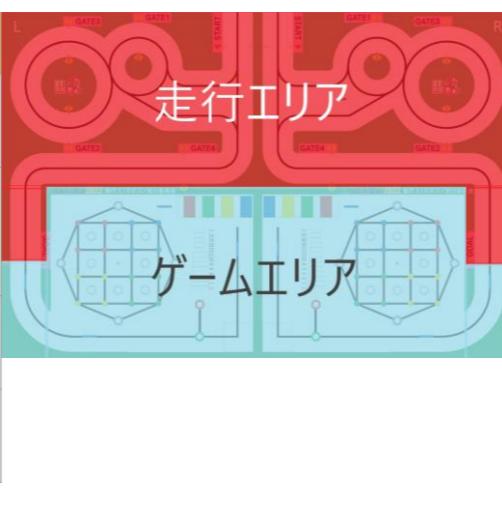
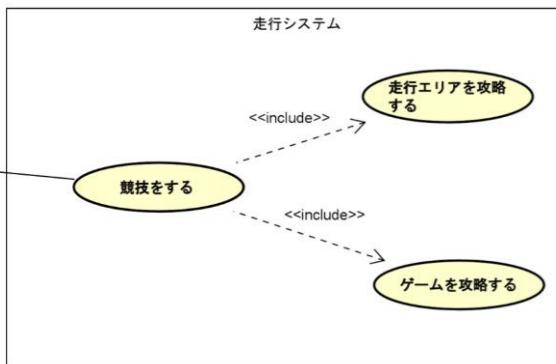


図1-1

1-2. ユースケース分析

開発の目標を達成するために必要な機能をシステムのユースケース図を示す[図1-2]また、機能ごとに必要な要件を抽出し、対応する方針を検討

図1-1 ユースケース図

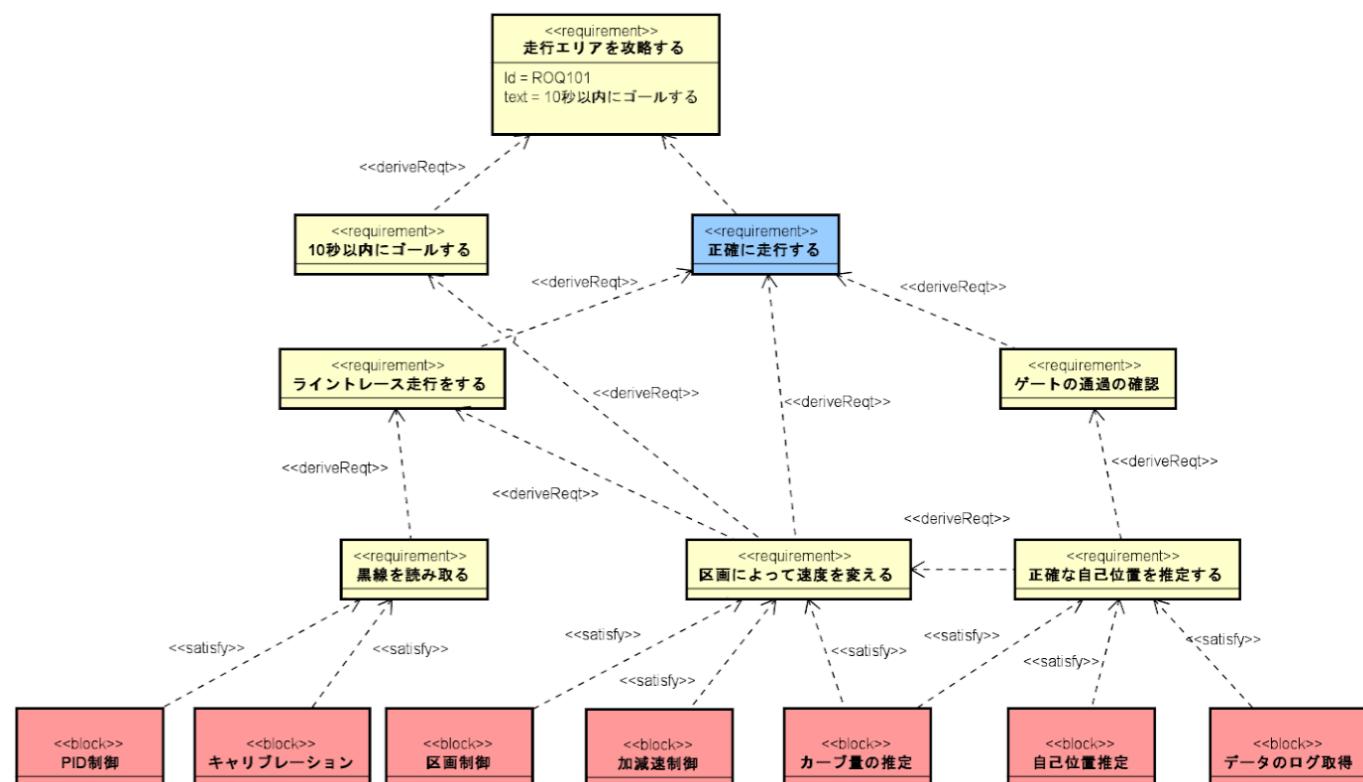


ユースケース名	競技をする
アクター	競技者
事前条件	走行体がスタートエリア内で停止している
事後条件	走行システムが正常に停止する
基本フロー	①競技者は走行体をスタートさせる ②走行システムは走行エリアを攻略する ③走行システムはゲームを攻略する ④走行システムは競技終了を通知する
例外フロー	強制的にユースケースを停止する

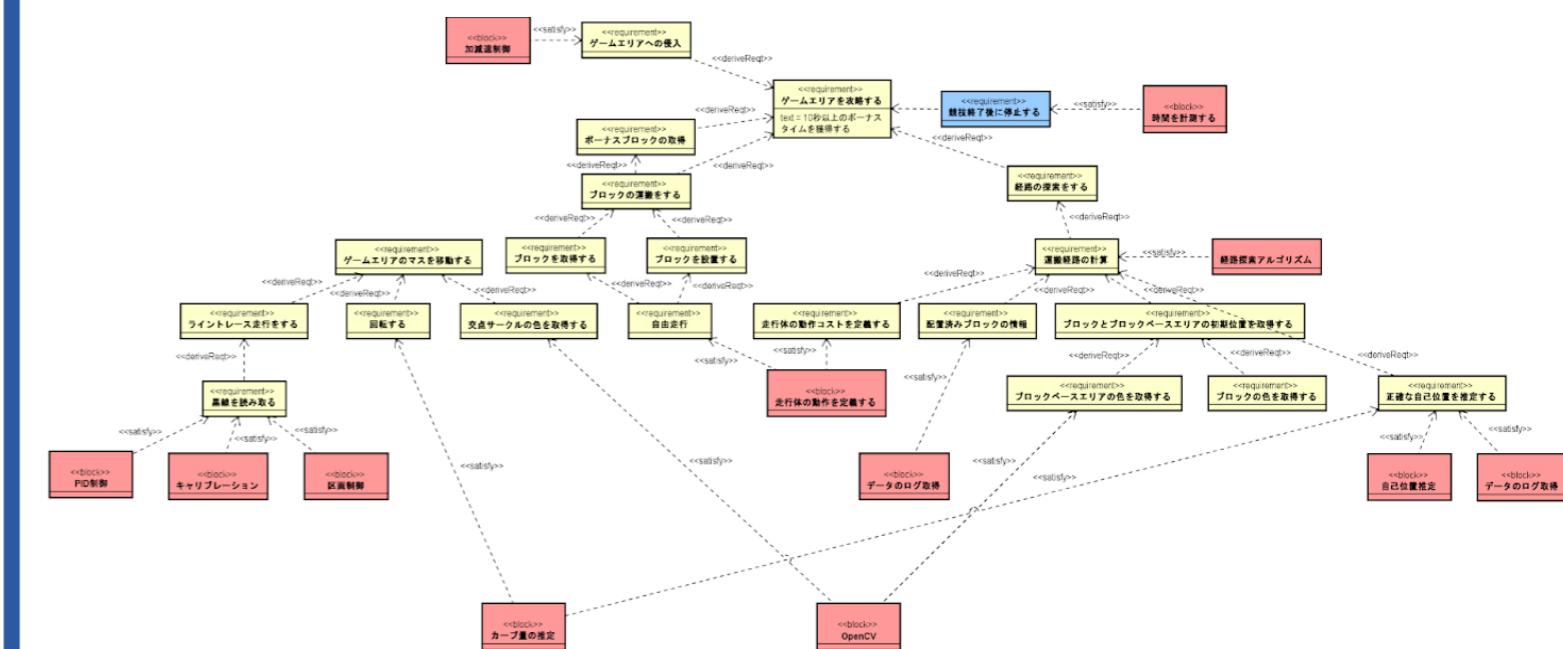
1-3. 要件定義

要求に対し満たす必要なシステムや技術を洗い出し、各要件に対応する関連を検討する。ユースケース分析から走行エリアとゲームエリアに分けて要件定義を行った。

- 走行エリアの要求図 -



- ゲームエリアの要求図 -



3. 設計モデル

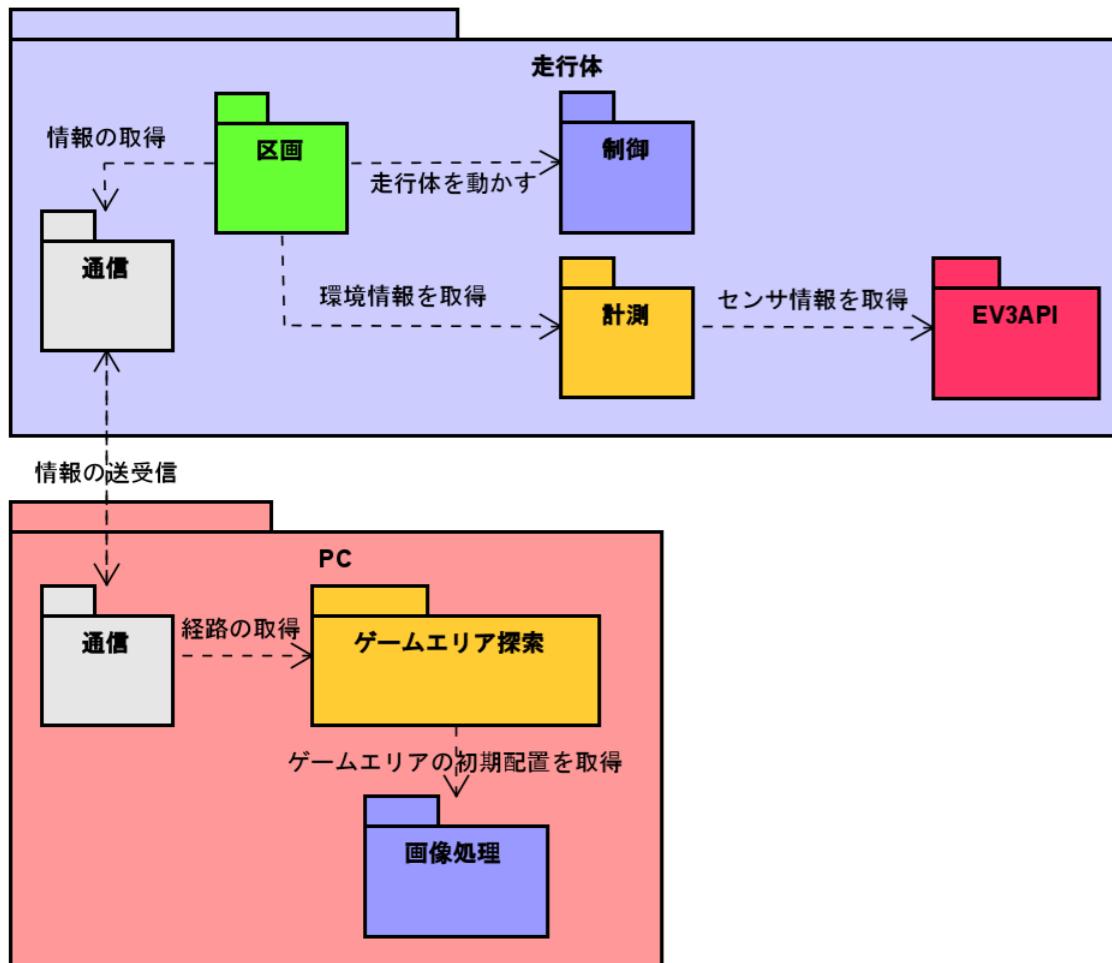


3-1.構造設計(全体)

要求モデルの分析結果からソフトウェア全体の構造設計を行い、それぞれ走行体側とPC側に分けた。
下記の表と図3-1 パッケージの構造に示す

デバイス	パッケージ	役割
走行体	通信	Bluetooth通信によりPC側との情報の送受信を行う
	区画	区画制御を行い状況に従って制御をする
	制御	EV3機能や走行体の動作を管理する
	計測	EV3APIの情報を管理し、データの加工を行う
	EV3API	各種モータ・センサ
PC	通信	Bluetooth通信により走行体側との情報の送受信を行う
	ゲームエリア探索	早く攻略するため、経路探索を行う
	画像処理	各種ブロックの座標及び色を推定する

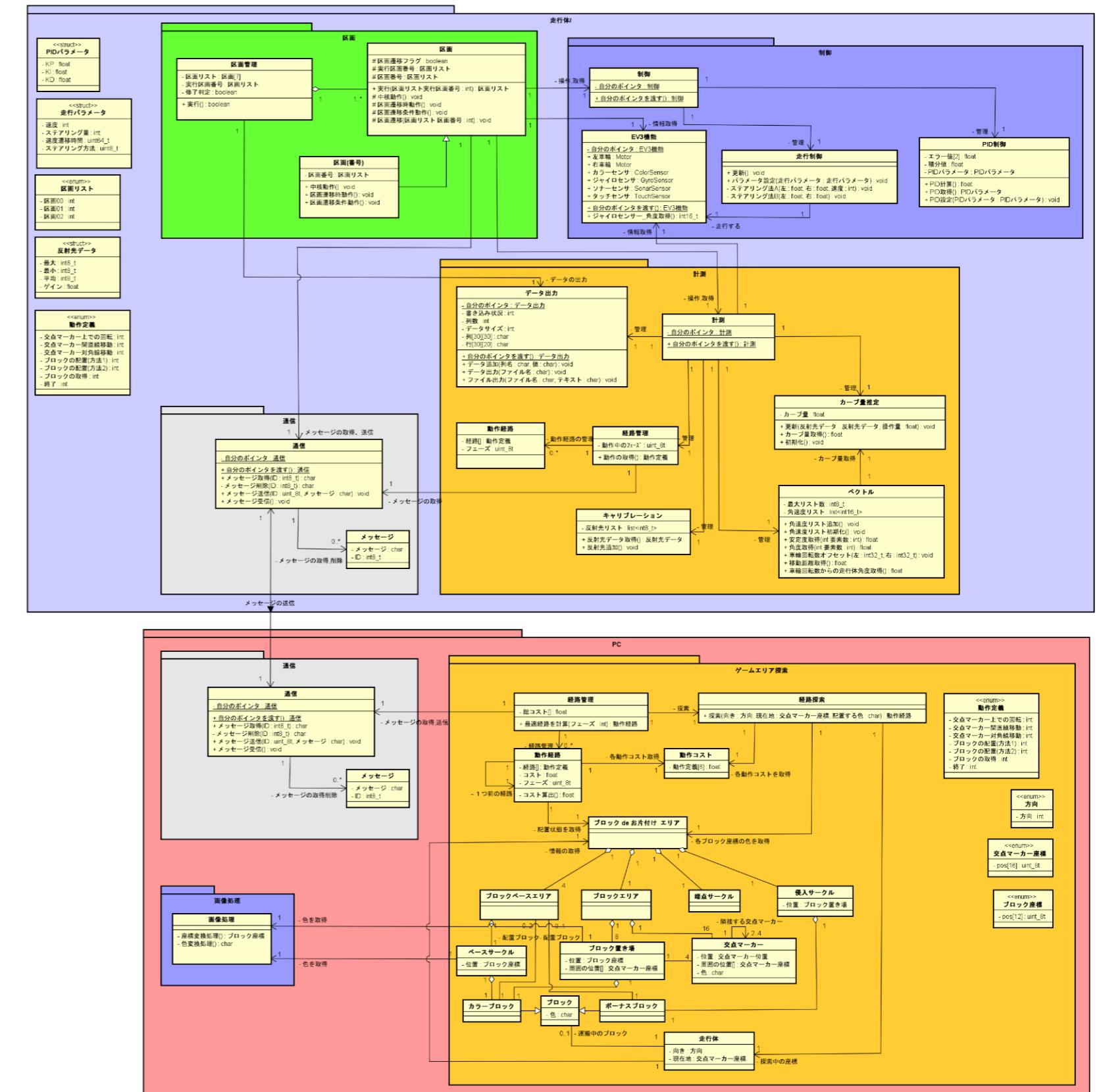
図3-1 パッケージの構造



3-2.構造設計(詳細)

3-1で定めたパッケージの構造を元にパッケージ内の詳細な設計を図3-2 パッケージの詳細設計に示す。また、区画制御の流れやパッケージ間の振舞に関しては次のページの図3-10から図3-13を参照

図3-2 パッケージの詳細設計



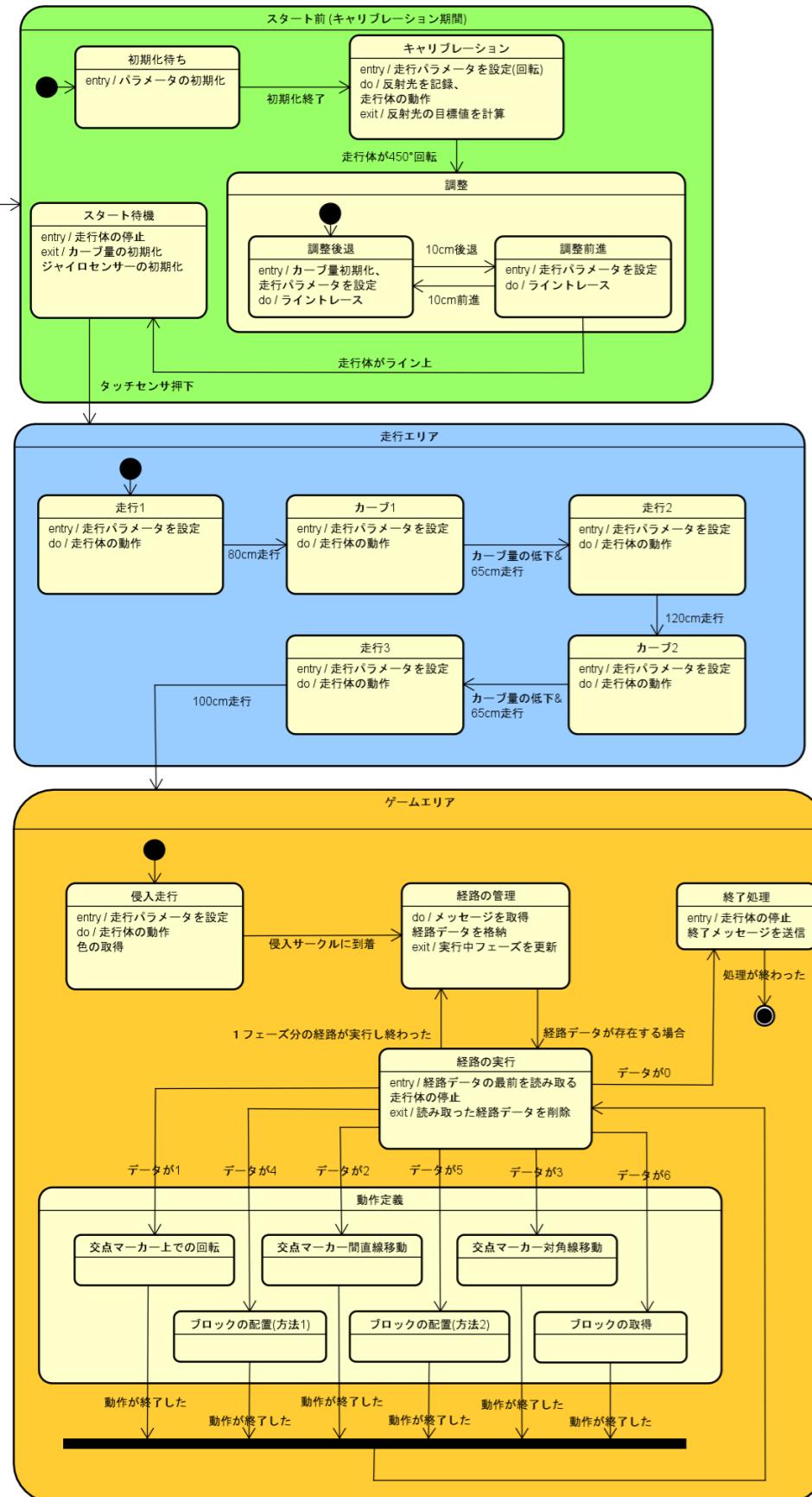
3. 設計モデル



3-3.振舞設計(詳細)

走行体の区画制御による遷移状態を下記の図3-10のステートマシン図で表す。

図3-10

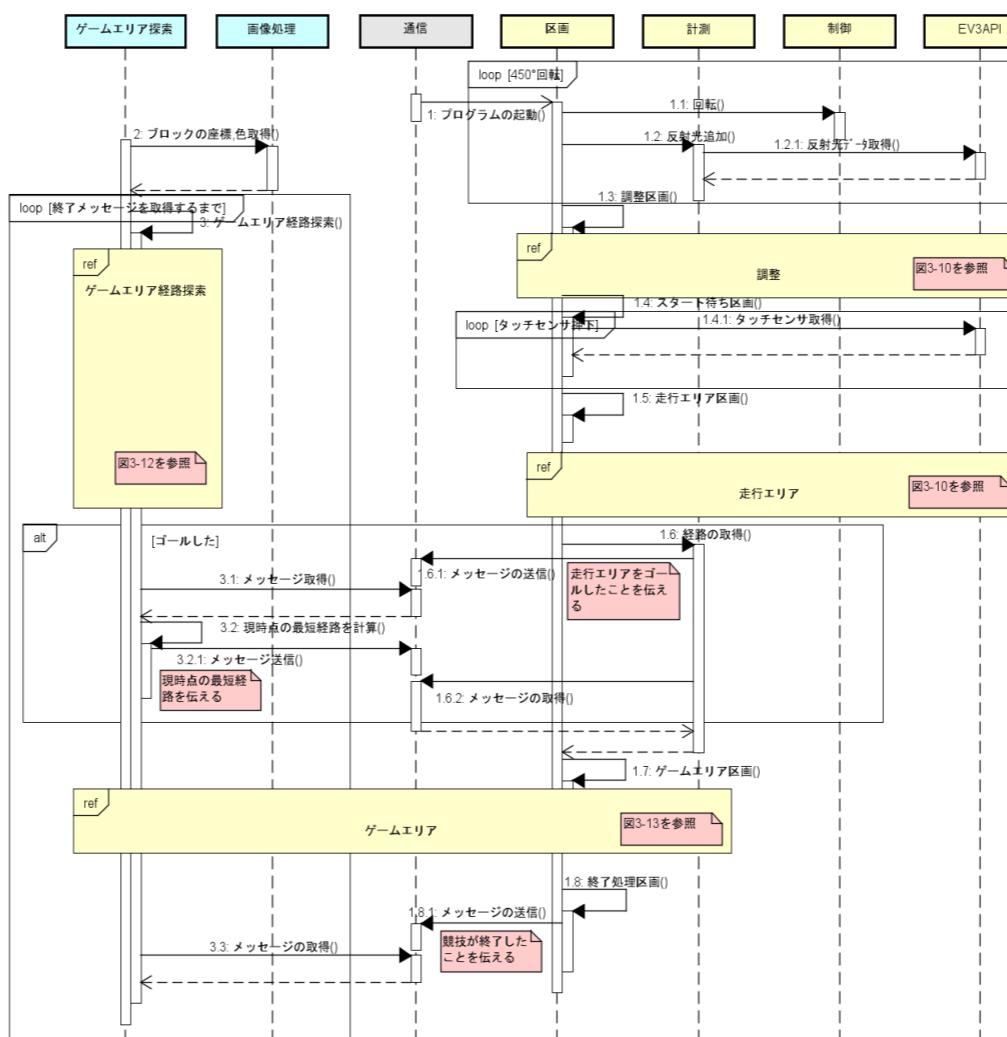


競技の開始~終了まで各パッケージごとの全体的な振舞を図3-11に示す。

ゲームエリア攻略中のPC(ゲームエリア)と走行体(区画)との通信を図3-12 ゲームエリアに示す。

ゲームエリア内のブロックまでの経路~ベースサークルまでの経路(1フェーズ)を探索する
ゲームエリア経路探索 図3-13に示す。

図3-11



4. 制御モデル

4-1.走行エリア

課題：走行体がカーブにさしかかっているかどうかを判断することが難しい

対策①：カーブ量を推定する機能を実装

走行体の各種センサーから独自の計算方法を用いてカーブ量を推定した。

計算方法は下記に示す。

$$a = (\text{反射光} - \text{目標値})^2$$

$$b = a * \text{操作量} * \text{期間角速度(gyro)}$$

$$C(x) = b + C(x-1) * 0.9$$

↓

$$C(x) = (\text{反射光} - \text{目標値})^2 * \text{操作量} * \text{期間角速度} + (C(x-1) * 0.9)$$

※操作量 = PID制御により生じた補正量

※期間角速度 = ある一定期間のジャイロセンサーの変化の平均値

計測したグラフからカーブにさしかかっている場合他の方法に比べ変化が大きかったが、過去の影響も含めて計算しているので反映に時間がかかるてしまいカーブの判定が遅ってしまった。

対策②：一定量走行した後カーブ走行に遷移する

カーブの判断を諦め、一定時間走行したときカーブ走行に遷移することにした。

直線走行に遷移する条件にカーブ推定を用いることによりうまくゴールまで走行することができた。

4-2.OpenCVによる色の分析

課題：ゲームエリアを攻略するためには各種設置ブロック、交点マーカーの色取得をする必要がある

対策：その手法としてOpenCVによる色の分析を利用

四色のブロックを撮影した画像を集め、機械学習を行った

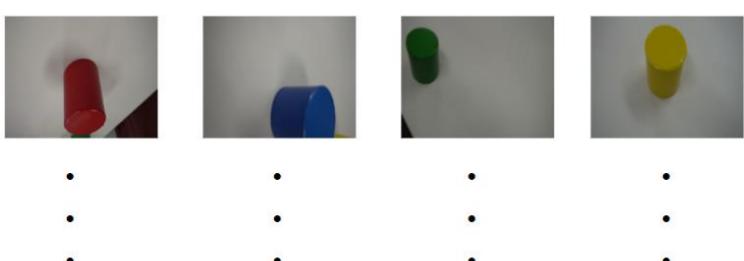


図4-1学習データの例

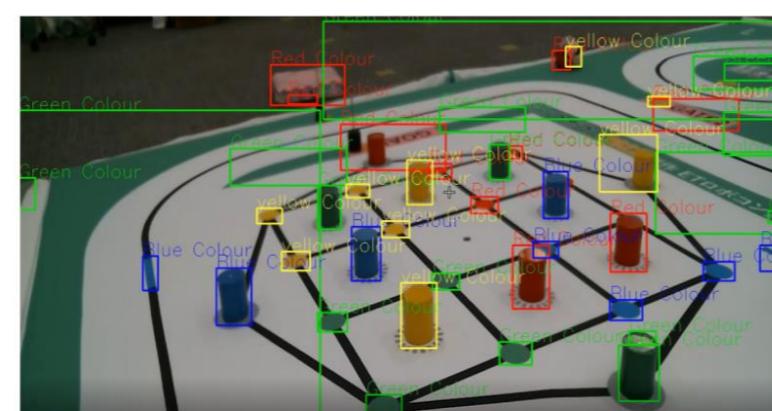


図4-2Opencvによる色の分類

集めたデータでk-meansとSVMを利用した
教師なし学習を行った

課題：緑色のブロック、交点マークの認識時、ゲームエリアの地面の緑色を認識してしまう。また、本番ではスポンサーの表示があるため、それを誤認識してしまう危険がある。

対策：キャリブレーション中に色を認識する範囲の選択をできるようにした。



図4-3Opencvによる色の分類

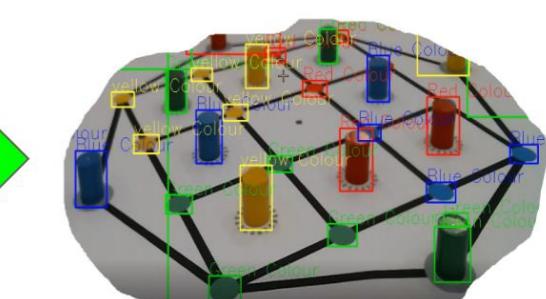


図4-4範囲選択後の画像

課題：認識した色の位置を初期位置に当てはめるのが難しい

対策：射影変換をしてブロックと交点マークの配置を単純化した。

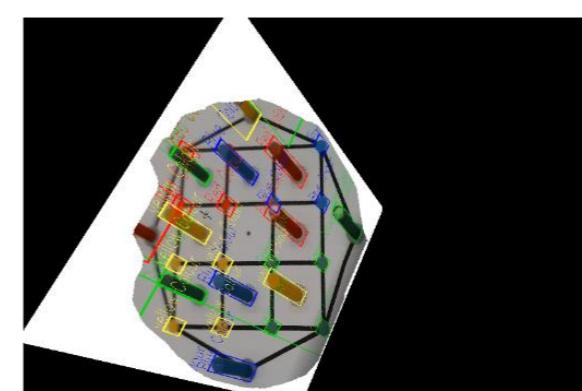


図4-5射影変換後の画像

4-3.経路探索

課題：ブロックベースエリアを走行時、ブロックベースエリアに設置済みのブロックに車体がぶつかってしまう。

対策：ブロック配置後、配置したブロックに隣接する直線を通行不可にする

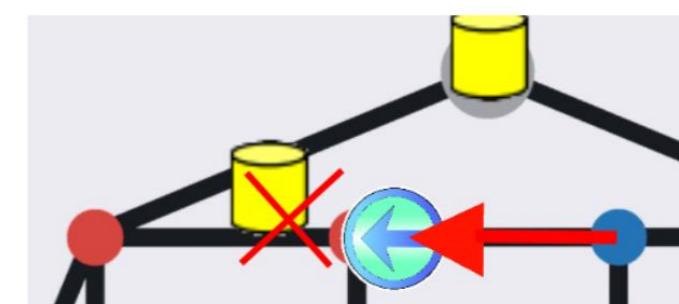


図4-6直線使用不可の例