仮想ロボットのトレーニングを学ぼう

レッスン 3：自動運転車

UNREAL ENGINE

生徒向けガイド

目次

仮想ロボットのトレーニングを学ぼう
レッスン 3：自動運転車

目的

このレッスンは前回の「レッスン 2：相撲ロボット」と類似していますが、新しい目標を達成するために類似した概念をどのように利用できるかを示します。また、Unreal Learning Kit で後で出てくる拡張アクティビティや将来のアクティビティを試す前に生徒の自信を高めることにもなります。

このレッスンでもセンサーを扱います。具体的には、受け取ったセンサー データに基づいてロボットがどのように判断するかを示します。

生徒は配送ロボットを作成し、目的地までのラインに追従するようにプログラミングします。拡張アクティビティでは、2 つのセンサーを持つロボットを構築することで、学習したことを広げます。わずかの追加コーディングで、ライン追従の動きをより滑らかでより正確にすることができます。

はじめに

入門編ガイド

Unreal Engine を初めてインストールおよび使用する場合は、このアクティビティに進む前に「入門編ガイド」を完了してください。このガイドには、このアクティビティをうまく行えるよう、Unreal Learning Kit のファイルをインストールする手順が記載されています。

「入門編ガイド」はこちらにあります。

レッスンの目的

すべての生徒はこれらすべてのレッスンを順序どおりに進めることをお勧めします。そうすることで、Unreal Learning Kit の全体像、および Unreal Engine 環境でロボット制作とモーター/センサーのコーディングの両方がどのようにサポートされているかを把握できます。

Unreal Learning Kit では、応用物理を使用してロボット工学の概念を探ります。また、生徒にコーディングに関するフィードバックを速やかに与え、今後扱う可能性がある他の物理ロボット システムに応用できる原則やコーディング言語の知識を提供します。

レッスン 3：自動運転車

はじめに

Tesla の工場の組み立てロボットが経路に追従する (「How the Tesla Model S is Made | Tesla Motors Part 1 (WIRED)」を参照) のを見たことや、自動運転車に乗ったことがありますか。これらのロボットはどのようにして道路や経路を見て、外部からの指令なしにそこを外れずにいるのでしょうか。配送ロボットを作成することで、自律走行車両の世界を探ってみましょう。

ライン追従ロボットは、素晴らしい自律走行車両の世界への入り口です。この簡単な実習では、車両が状況を利用して自主的にナビゲートできる一つの方法を紹介します。ライン追従ロボットの仕組みを理解するには、光センサーが何を検出し、その情報をどのように利用して事前設定されたラインや経路の上や内側に留まっているかを理解しなければなりません。

このアクティビティでは、光センサーを使用してラインに追従するロボット、つまりライン追従ロボットをコーディングします。

レッスンの概要

ロボットを制御するために、ロボットがガイドラインから離れ、再びガイドラインの方に引き返し、最終的にラインに追従して前進し続けるようにコーディングする方法を学びます。

ロボットはラインを見つけると、そのラインから少し離れます。そのラインが見えなくなったら、逆方向に引き返して、もう一度そのラインを探して走行し続けます。このプロセスを連続的に繰り返して、ロボットを実質的にライン「上」に留まらせます。このラインに沿って走行するときに、ロボットにはちょっとした「小刻みな動き」をします。

コードには、条件文が含まれたループがあります。この条件文では、センサーが結果をテストし続けて、ラインの方に向きを変えるかラインから離れるように向きを変えるように絶えず調整を行います。
光センサーを使用するには、次のことを理解しておく必要があります。

- 光センサーはロボットに数値を返します。
- 条件文では、センサーの現在値をテストして、その数値が低い範囲内 (黒いサーフェス = ライン) と高い範囲内 (明るいサーフェス = 床) のどちらであるかを判断します。
    - 値が低い範囲内であれば、ロボットにはラインが見えていることを意味するので、ロボットがラインから離れる方に走行するようにプログラミングします。
    - 同様に、値が高い範囲内であれば、ロボットには床が見えていることを意味するので、ロボットがラインの方に向けて走行するようにプログラミングします。
- そのコードは、連続してループする必要があるので、センサー値をテストしてモーターを制御することを 1 秒間に数回行います。
- ロボットは目的地で停止します。このアクティビティの現行のロボット モデルは、目に見えないブロッキング ボリューム ボックスにぶつかると停止するので、この行動をコーディングする必要はありません。

このアクティビティの拡張では、2 つ目のセンサーによって、ロボットが前進するときにもっと滑らかな動きをどのようにして作り出すことができるかを探ります。

コーディングのつながり

このアクティビティを完了すると、生徒は実践的な環境でのループと条件文の活用法を理解できるようになります。このロボットには、明暗の違いを検出するセンサーが備わっています。返されたセンサー値をしきい値と比較して、走行中に白い枠線 (地面) と黒いライン (経路) のどちらが見えているかを判断します。

条件文は、ロボットがラインに追従することに役立ち、ループは目的地に到達するまでロボットが連続的に動くことに役立ちます。

ラインを見つけましょう!

レッスンを開始する

ロボットの構築;光センサーを使用する

前回のレッスンでは、光センサーを使用して、ロボットが相撲の土俵にある境界線を確認できるようにしました。配送ロボットに対しても同じ方向性を使用しますが、同じセンサーを使用して、経路に沿ってロボットを案内できるようにコードを変更します。

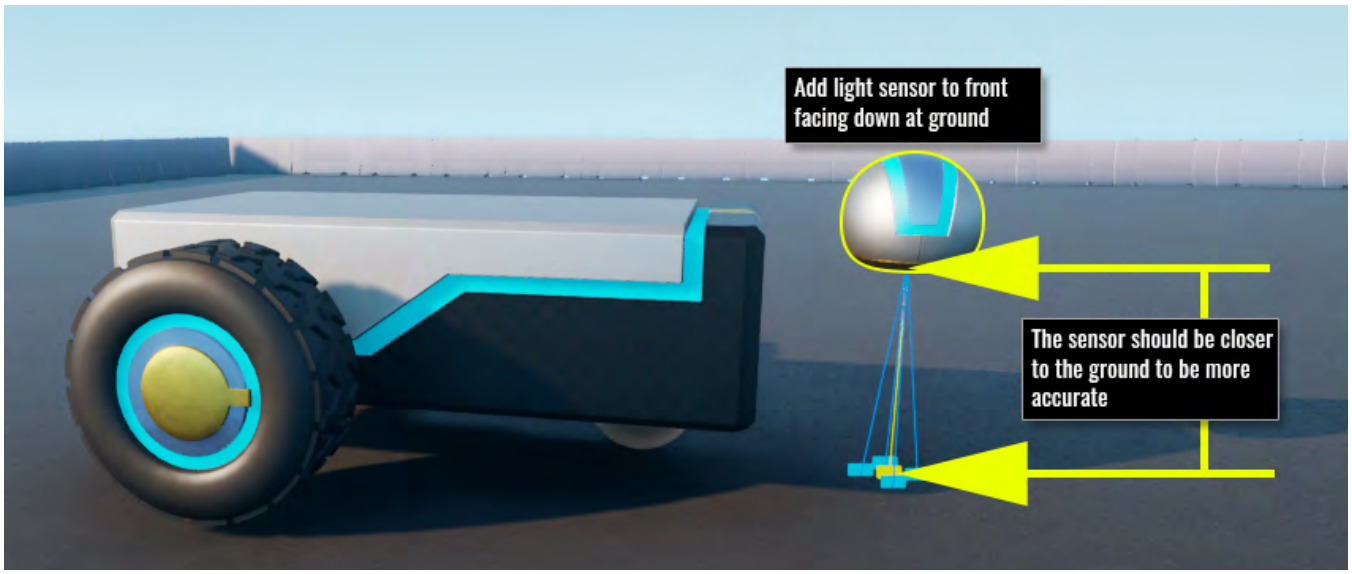


図 3-1

目的を定める

ここで使用するロボットの特定の役目は、地面上のラインに追従することです。環境に対する要件を特定し、役目を完了するために従わなければならないルールを定めます。

環境

- ロボットは走行しなければならない方向を向いているライン上に配置される
- 追従すべきラインには、滑らかなカーブがあり、ライン自体は重なり合っていない
- 走行するサーフェスは平らである

ルール

- ロボットはユーザーの介入なしで、コードによって自律的に走行しなければならない
- ロボットは直ちに走行を開始し、ラインの終点に到達するまで走行し続ける
- ロボットがラインを見失ったら、ロボットを停止して修正してからもう一度試行する

要件

- センサーを使用して黒いライン (経路) と白い枠線 (地面) を「見る」
- 黒いラインが「見えている」ときはラインから離れるように走行する
- 白い枠線 (地面) が「見えている」ときはラインの方に向きを変える

実習

実習 1-a：ライン追従マップを開く

このアクティビティでは、ラインの開始位置に 1 つのスターター ロボットがいるプロジェクト ファイルを開きます。このロボットは機能しないので、センサーを追加し、動作するようにコーディングする必要があります。まず光センサーをロボットに追加し、センサー値をテストし、ロボットをいつ動かすかを判断する追加コードを作成します。

- コンテンツ ブラウザで、「Content」->「LearningKit_Robots」->「Maps_Robots」フォルダに移動します。
- 「L3」フォルダをダブルクリックしてを開きます。
- そして、「Map_3-1_FollowLine」をダブルクリックして開きます。
- ロボットの前にカーブした白地に黒の経路が広がり、その開始地点に 1 つのロボットが配置され、2 台の別の車両が駐車されている駐車場のそばが目的地として示されている、マップが表示されているはずです。

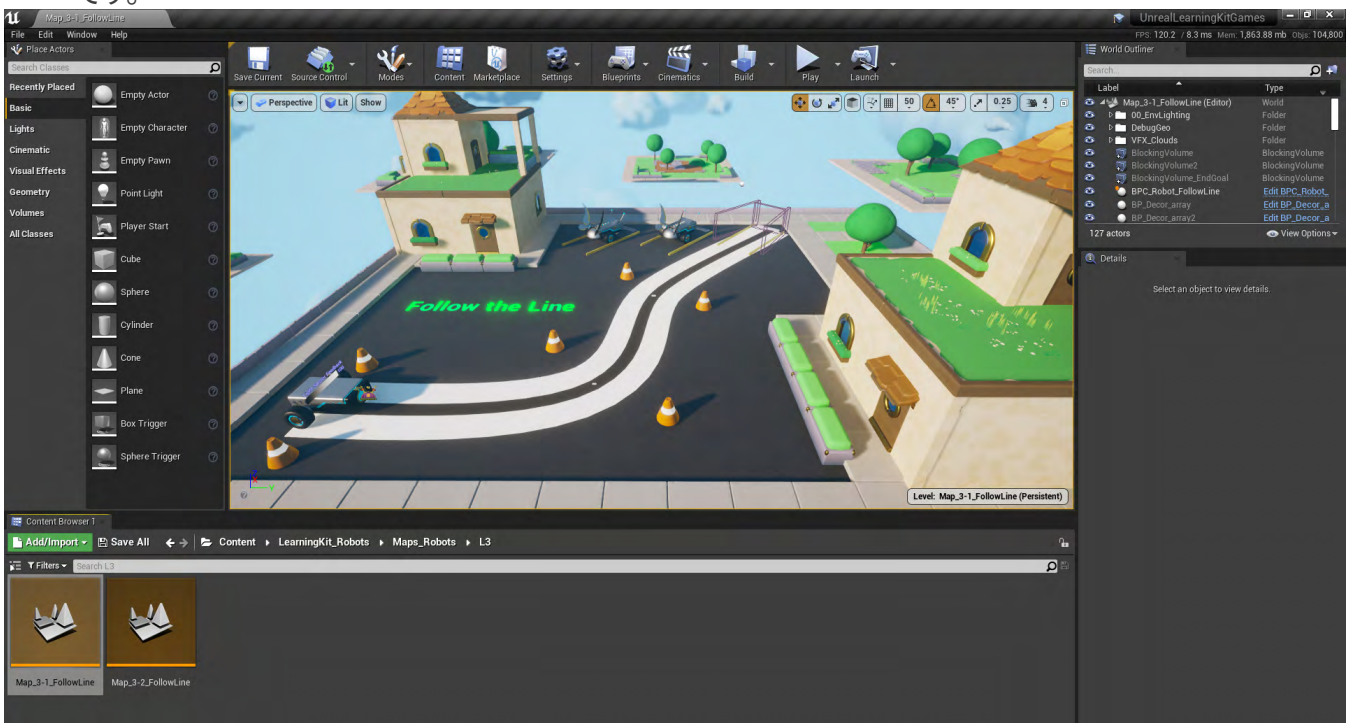


図 3-2

実習 1-b：光センサーをロボットに追加する

- このアクティビティでは光センサーを使用します。センサーの位置は、その正確さ、およびこれからコーディングするアクションへの対応に影響を及ぼします。

センサーを追加する前に、以下の状況を検討する必要があります。

- サーフェスからセンサーまでの高さは、センサーの正確さに影響を及ぼします。
- ロボットでのセンサーの位置が前から後ろであると、ロボットの転回行動に影響を及ぼし、その信頼性にも影響を与えます。

センサーの最適な位置を確認できるように、ブループリントのビューポートにヘルパーを追加しています。次の画像のような Cleverlike ロゴを探してください。

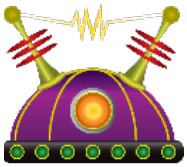


図 3-3

ロボットを編集する

- ビューポートにあるロボットをクリックして選択します (画面の右側にある [Details] パネルおよびアウトライナで、アクティブなアクタとして BPC_Robot_FollowLine が表示されているはずです)。

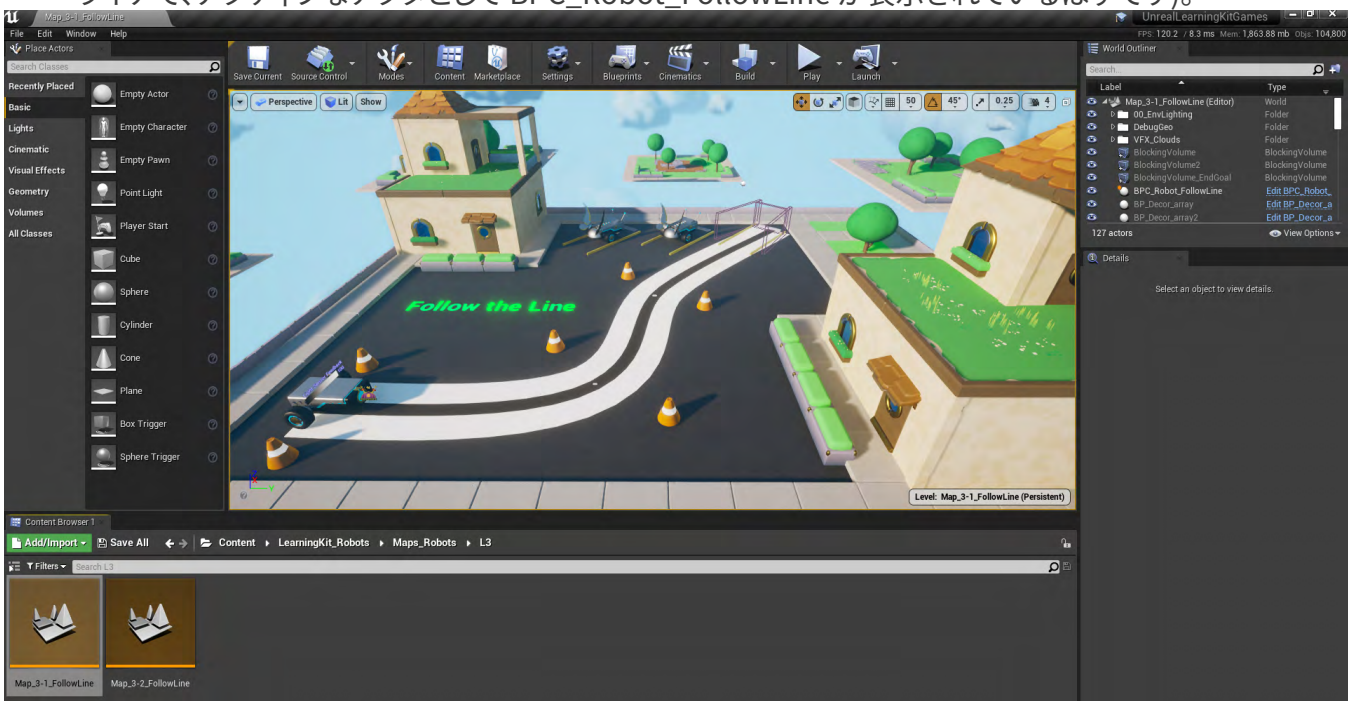


図 3-4

- **[Details]** パネルで次の手順を行います。**[Edit Blueprint (ブループリントの編集)]** をクリックし、**[Open Blueprint Editor (ブループリント エディタを開く)]** をクリックします。
- **BPC_Robot_FollowLine** ブループリント内に**イベント グラフ**が表示されているはずです。

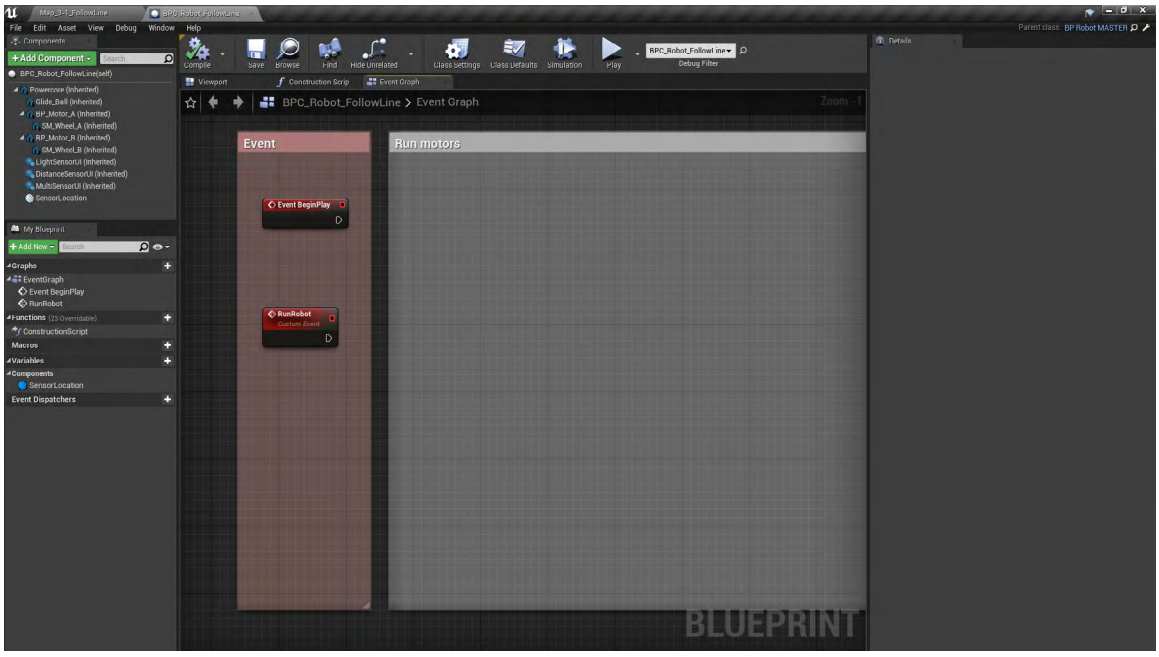


図 3-5

- 実習 3-1 では、左側の条件ボックスにある **Event BeginPlay** ノードと **RunRobot** ノード以外に、ここにはコードがなく、右側の Run Motors ボックスにもコードがないことに注目してください。

光センサーを追加する

- インターフェースの左上にある **[Components (コンポーネント)] パネル**で、緑色の **[Add Component (コンポーネントを追加)]** ボタンを**クリック**して、検索バーに「sensor」と入力します。
- **BP Sensor Light** をクリックして選択します。

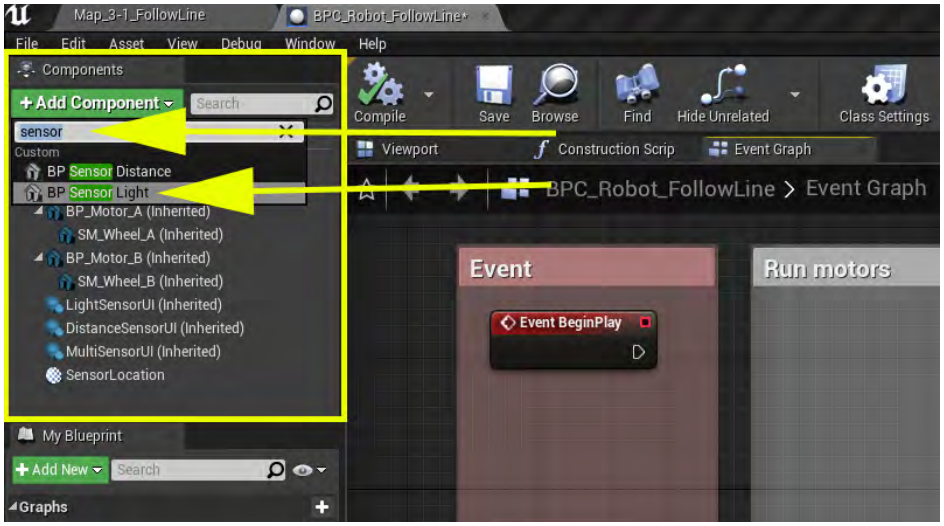


図 3-6

- そうすると、コンポーネント一覧の一番下に新しいコンポーネントが追加されます。
- 新しいコンポーネントが **LightSensorUI** コンポーネントの直下に表示されていない場合は、別のコンポーネントに追加した可能性があります。その場合は、**BP_Sensor_Light** をクリックして、それがロボットの Powercore に従属するように、**Powercore までドラッグ**します。そうすると、それがコンポーネント一覧の一番下に表示されます。それが別のコンポーネントの中にある場合、Powercore (ロボット) の子ではなく、そのコンポーネントの「子」になります。

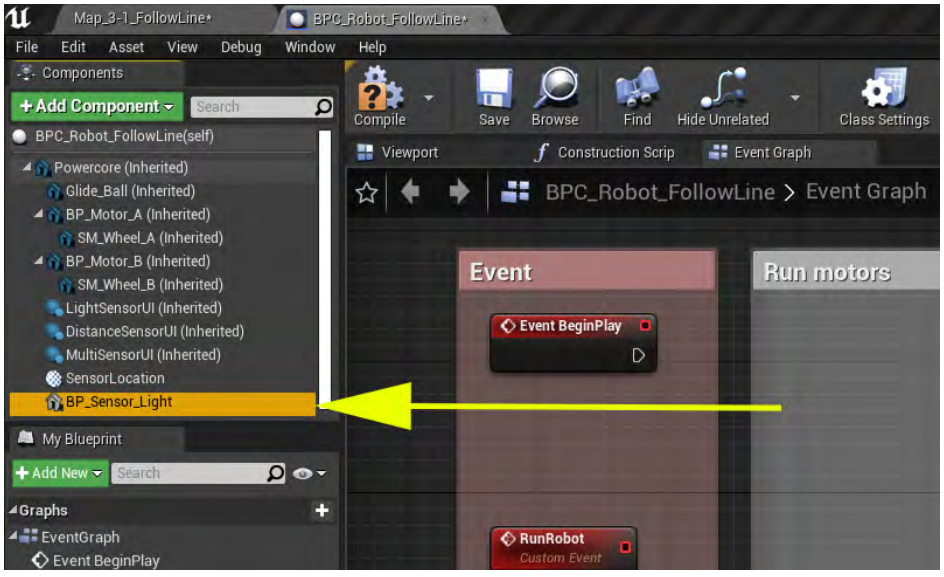


図 3-7

- センサーが **Powercore** を基準としてどこに配置されているかを確認するには、**ビューポート** タブをクリックします (図 3-8 を参照)。

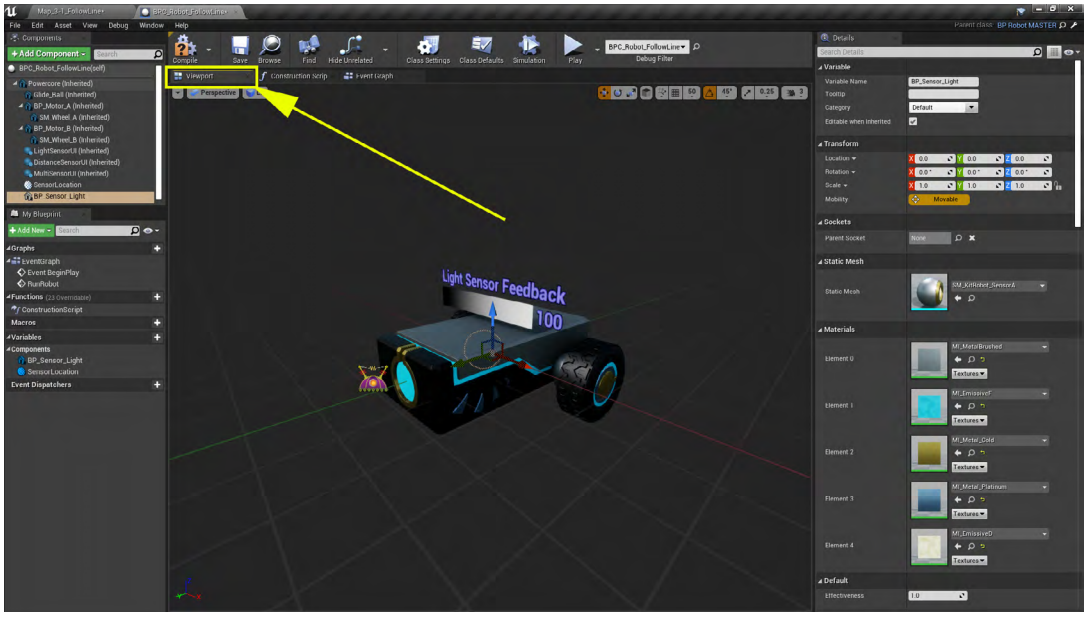


図 3-8

- ヘルパーが表示されるまで、ロボットの向きを変えます。**プロのアドバイス：F** キーを押すと、選択されているロボットにズームインして、それが中央に拡大表示されます。
- センサーは自動的に **Powercore** 内に配置されています。それが、ロボット内で薄黄色の形状として選択されています。このセンサーを、最も効果的に機能する適切な位置に動かす必要があります。そうするには、**ビューポート** ウィンドウの左上にある動くツールを使用します。
- 1 つ目のツールをクリックして、オブジェクトの選択と平行移動ツール (Move Gizmo) を使用するか、または **W** キーを**押します** (ギズモが三方向矢印として表示される)。このギズモの矢印をクリックしてドラッグし、光センサーが Cleverlike ロゴを覆うようなるまで、センサーを上下左右に動かします。これが初期位置です。
-

- センサーは横を向いたままなので、地面を見下ろすようにセンサーの向きを変える必要があります。
- この場合も、**ビューポート** ウィンドウのツールバーの右側にある 2 つ目のツールをクリックするか、または **E** キーを押して、オブジェクトの選択と回転ツール (Rotation Gizmo) を使用して、地面を向くように向きを変えます。

- 右側にある **[Details] パネル**の **Transform** セクションで、選択しているオブジェクト (光センサー) の設定が変化しているのがわかります。Move Gizmo (W キー) で Location 設定を変更し、Rotate Gizmo (E キー) で Rotation 設定を変更しています。センサーを上下にドラッグすると、**[Details]** パネルにある Location Z ボックスの値が変化します。
- センサー付きのロボットは次の画像のようになります。

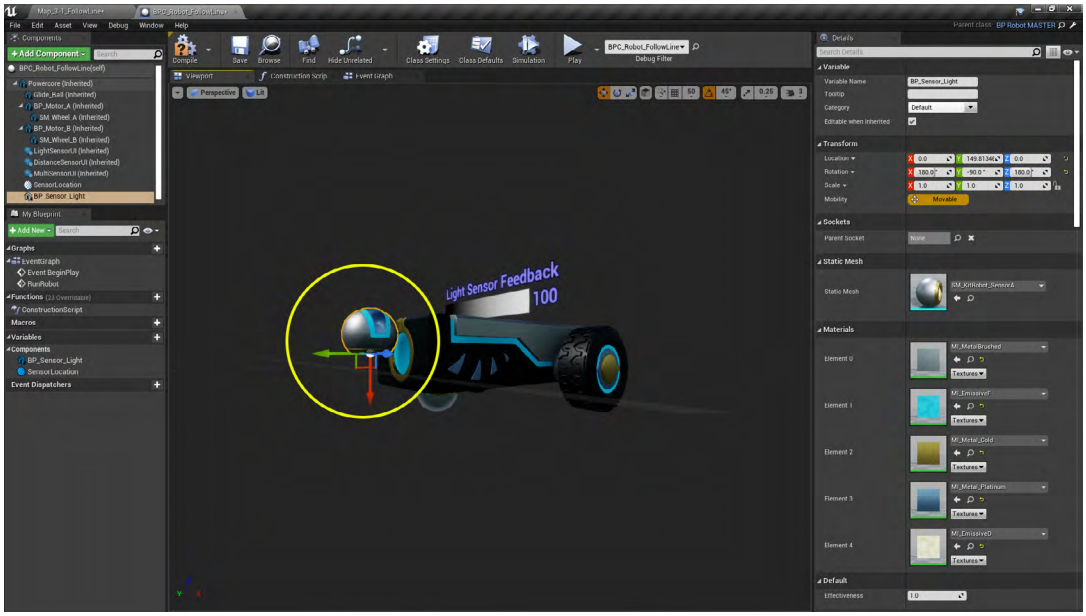


図 3-9

- **BP_Sensor_Light** を選択した状態で、右側の **[Details] パネル**に注目します。その方が都合がよければ、Location フィールドに値を入力することもできます。

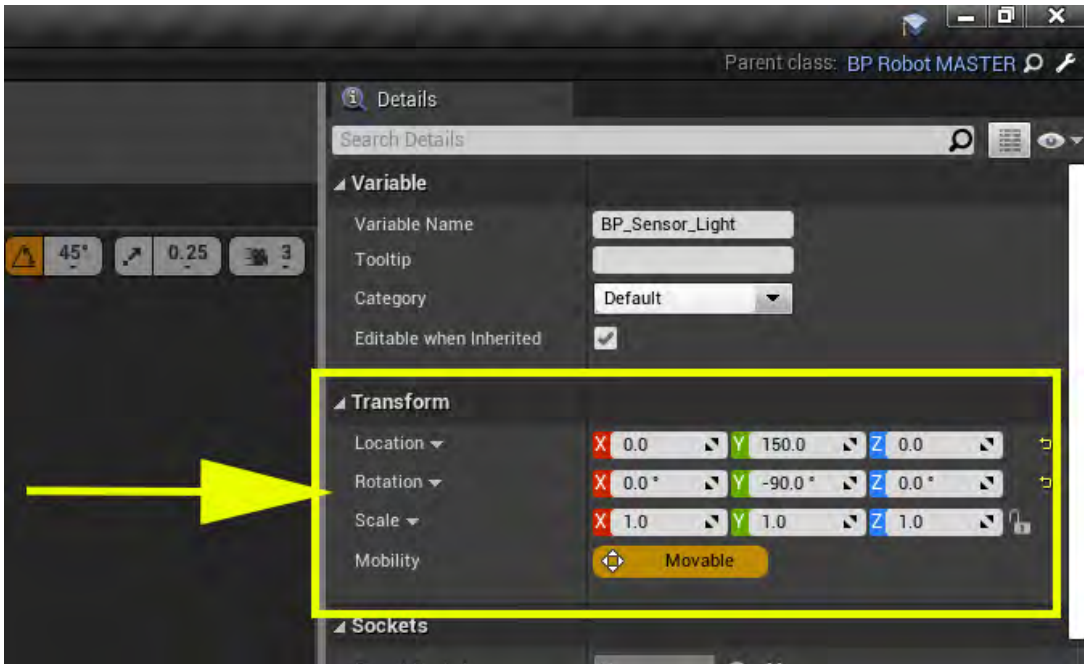
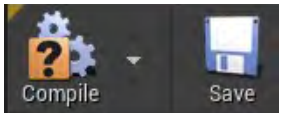


図 3-10

- 最後に、ロボットの内部コードでの変更内容をコンパイルする必要があるので、ツールバーの**左上にある [Compile (コンパイル)] をクリックしてから、**[Save (保存)] をクリックします。
- *(コンパイルして保存すると、[Compile] ボタンがオレンジ色のクエスチョン マークから緑色のチェックマークに変わります)。*



画面の上部にある **Map_3-1_FollowLine** タブをクリックしてレベルに戻り、ロボットに**ズームイン**します (マウス ホイールによるスクロールまたは F キー)。ビューポートで、ロボットの前面にセンサーが表示されているはずです。

図 3-11

- レベルにあるロボットとそのロボットが向いている方向に注目します。ロボットは前進する経路の方を向いていて、黒いラインの左側の縁の上にいる必要があります。そうなっていないと、ロボットはラインを適切に見つけることができません。
- 経路ライン上をロボットが前進する必要がある方向に、ロボットのビューを揃えます。

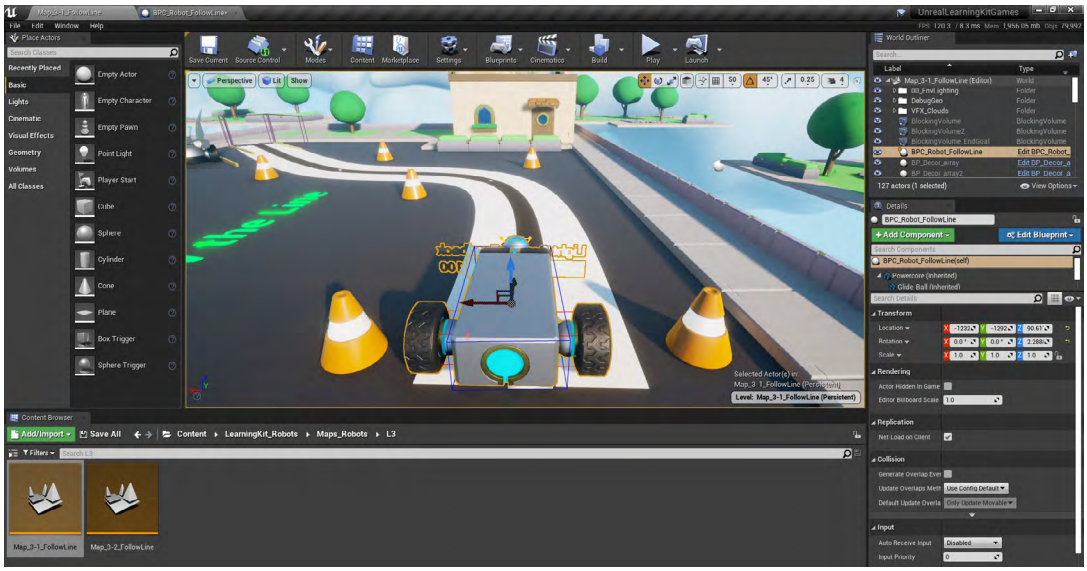


図 3-12

実習 2：センサーからの入力を取得する

このロボットは白い枠線と黒い経路ラインの違いを検出する必要があるので、光センサーを使用して、その違いを検出します。黒いラインに注目するので、2 つの条件式はラインが「見えている」とラインが「見えていない」になります。

センサーのタイプ

- 反射光を測定する光センサーを使用します。明るいサーフェスや白色のサーフェスは、暗いサーフェスよりも多くの光を反射するので、光センサーは明るいサーフェスでは大きい数値を返し、暗いサーフェスでは小さい数値を返します。

センサーの位置

- センサーは車両の走行中に地面を見下ろす必要がある
- 正確な測定値を得るために、センサーは地面の近くにある必要がある
- センサーはロボットの前面で車輪から離れた位置に配置される必要がある

光センサーのコードを追加する

センサーを理想的な位置に配置したら、センサーが動作して見ている入力を返すように、ロボットのコードにコマンドを追加する必要があります。ここでは、光センサーを使用するようにロボットに指示するために、光センサーのコードをブループリント エディタに追加します。

注記：Notepad または学校やクラス用の共有ドキュメント システムを使用して、以下の手順を進めるときに光センサーの値をログ記録します。

- レベルの**ビューポート**で、ロボットをクリックします。
- **[Details] パネル** (右側) を見て、BPC_Robot_FollowLine を開いていること確認します。

ブループリント エディタを開きます。

- **[Details] パネルで [Edit Blueprint] をクリック**し、**[Open Blueprint Editor]** をクリックします (ブループリント エディタ内に**イベント グラフ**が表示されているはずです。ブループリントのビューポートが表示されたままであれば、**イベント グラフ**タブをクリックします)。
- 今は**イベント グラフ**が表示されているはずです (図 3-13 を参照)。

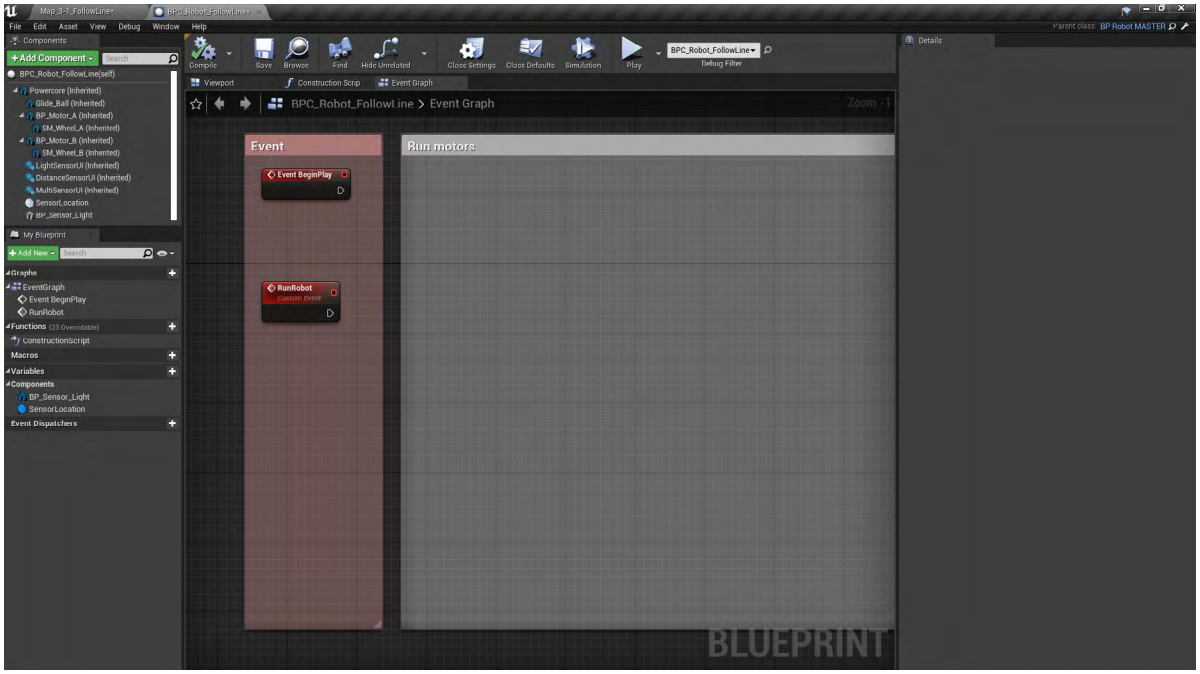


図 3-13

- 左側の**条件ボックス**にある **Event BeginPlay** ノードと **RunRobot** ノード以外に、ここにはコードがなく、右側の **Run Motors** ボックスにもコードがないことに注目してください。
- 左側の**イベント ボックス**で、**Event BeginPlay** ノードの出力実行ピンをクリックして右側に引き出し、灰色の **Run Motors** まで伸ばして、(白色の) ワイヤーを作成します。
- そうすると、実行可能なアクションの一覧がポップアップ表示されるので、必要な関数を検索できます。
- 「run robot」と入力して **Run Robot** 関数を検索します。

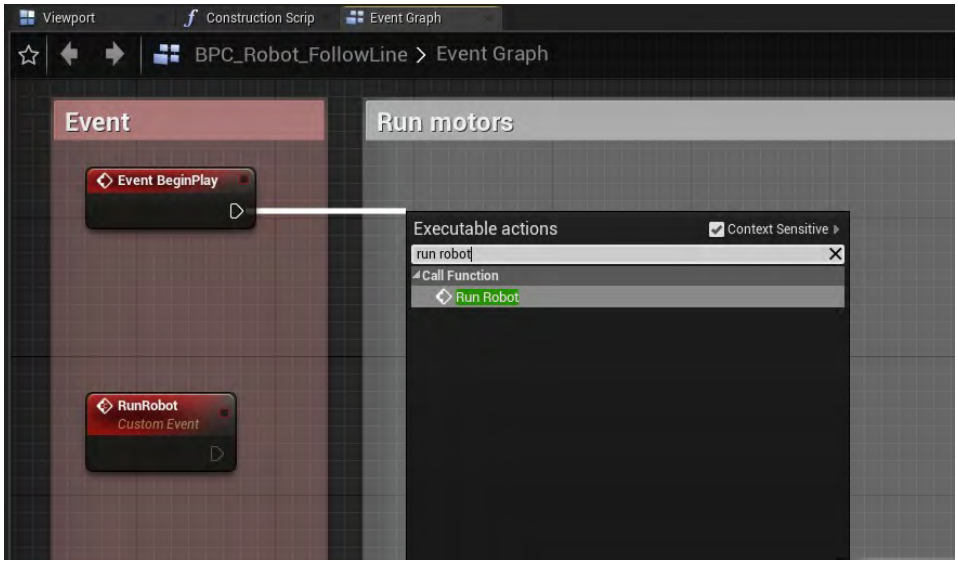


図 3-14

- **Run Robot** が灰色で選択されている状態で、キーボードの **Enter** キーを押して、その関数を選択します。そうすると、**Run Robot** ノードが追加され、出力実行ピンから、そのノードをつないでいる入力実行ピンまで、ワイヤー (白色の線で表示されている) が自動的に作成されます。これで、ゲームをプレイするときに **Run Robot** イベントが呼び出されます。

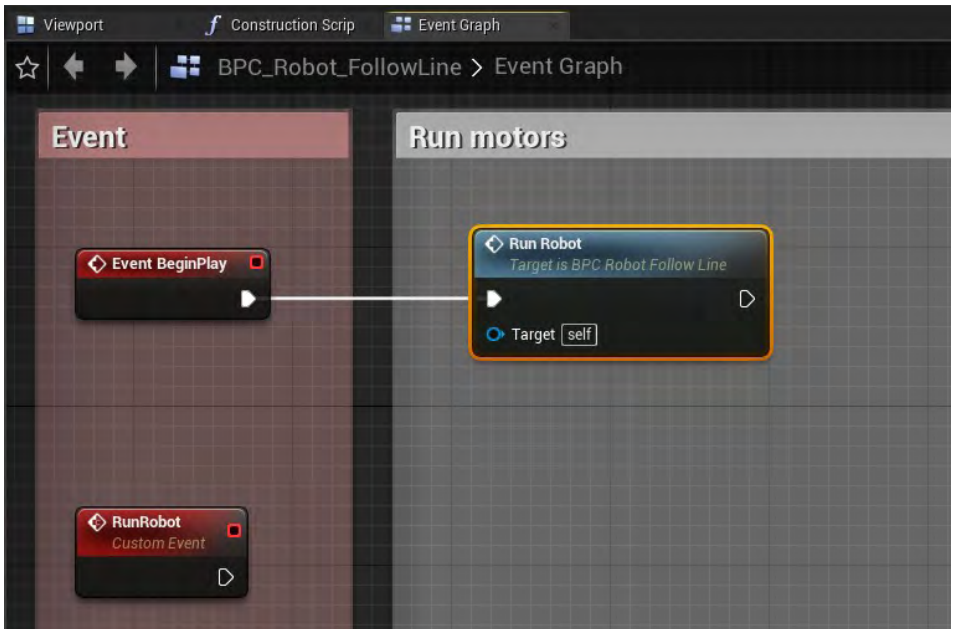
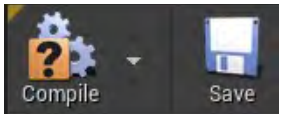


図 3-15

- コードを追加したときは毎回「コンパイル」して「保存」することを忘れないでください!



- *次に、**Run Robot** 関数が呼び出されたときにセンサーが実行されるように、センサーをつなぎます。**[Components (コンポーネント)]** パネルで、**BP_Sensor_Light** をクリックし、それを灰色の Run Motors ボックスにドラッグします (図 3-16 を参照)。*

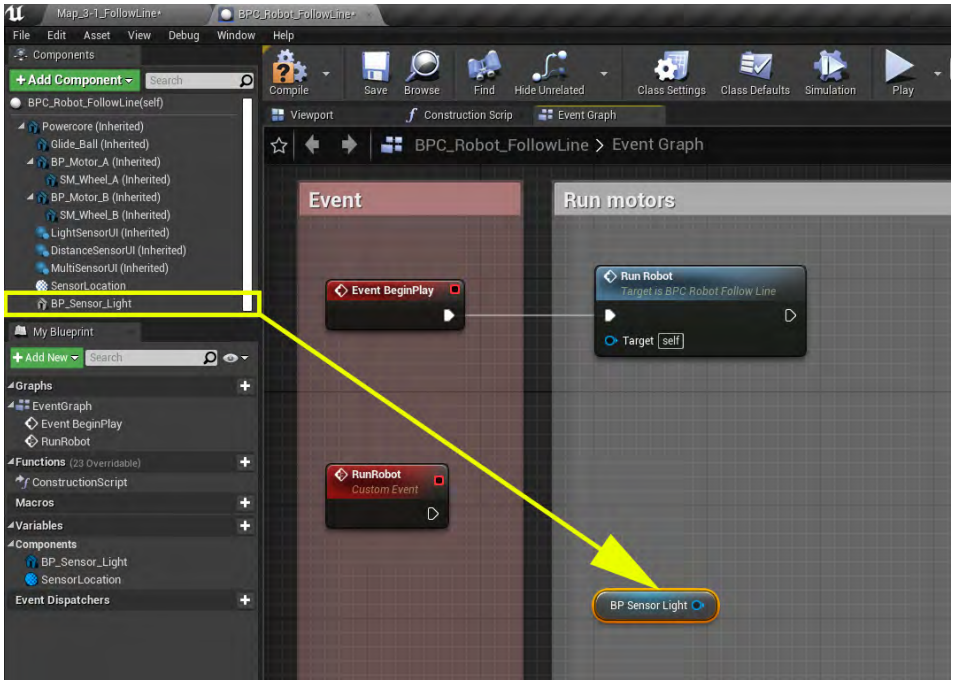


図 3-16

- *次に、センサーを、そのセンサーを実行するノードにつなぐ必要があります。青色の **BP_Sensor_Light** の出力ピンから右側に引き出して、灰色の背景の空き領域でマウス ボタンを離します。*
- *そうすると、実行可能なアクションの一覧がポップアップ表示されるので、必要な関数を検索できます。*
    - 「Run Light Sensor with Threshold」という新しい特化型ノードを使用することで、時間を短縮します。この新しいノードは **Run Light Sensor** ノードと似ていますが、条件文が組み込まれています。このノードが光センサーを実行するので、このノードにある **Threshold** の値を変更するだけで済みます (図 3 -18 を参照)。
- *ポップアップ ボックスに「un light sensor with threshold」と入力して検索し、**Run Light Sensor with Threshold** ノードを選択します。*
-

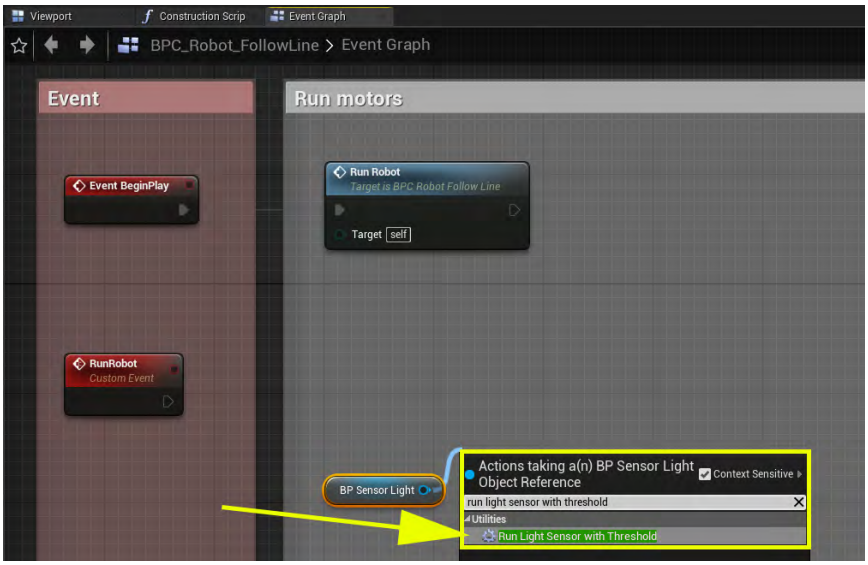


図 3-17

- *そうすると、**Run Light Sensor with Threshold** ノードが**イベント グラフ**内に配置され、その Sensor 入力ピンに **BP_Sensor_Light** ノードから青色のワイヤーがつながります。*

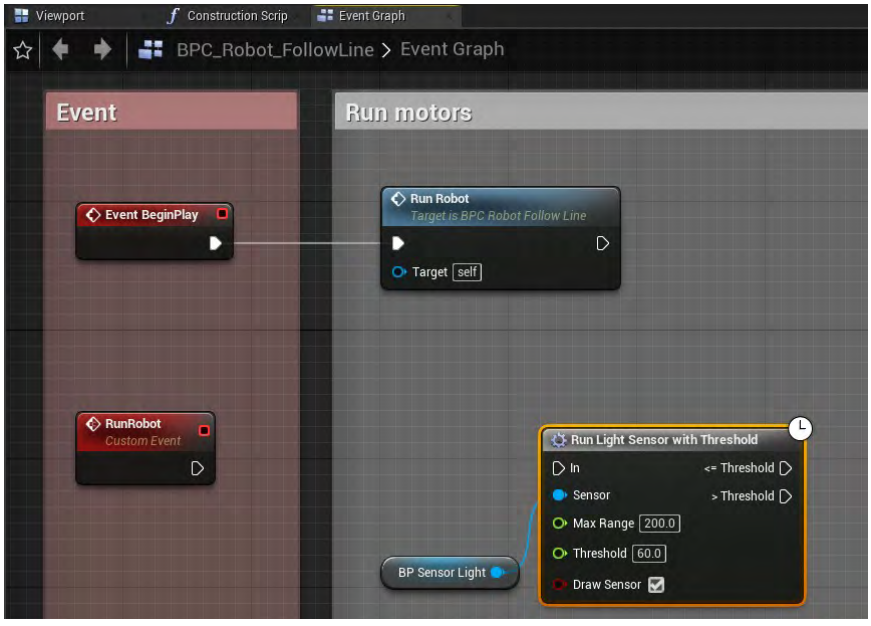


図 3-18

- *最後に、**Run Robot** ノードを **Run Light Sensor with Threshold** ノードにつなぐ必要があるので、**Run Robot** ノードの白色の実行ピンから **Run Light Sensor with Threshold** ノードの入力ピン (これも白色) につなぎます。*

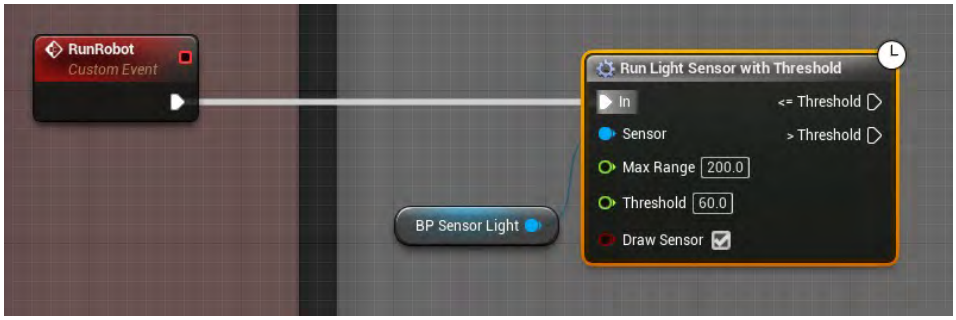


図 3-19

このプロセスでは、センサーのノードをコードに配置し、ゲームのプレイ時にそれを実行するように指示しました。ただし、現在のコードは、それらのコマンドを実行して直ちに停止するので、センサーが何を検出しているかを確認することはできません。センサーが何を検出しているかを観察するには、ループを作成する必要があります。

プロのアドバイス

ワイヤーを切断する必要がある場合は、Alt キーを押したままにしてそのワイヤーまたはピンをマウスで左クリックします。

必要に応じて、このセンサーが「見る」ために使用しているラインを確認する機能をオンにします。

- **Draw Sensor** の横にあるオプション ボックスをクリックし、ライントレースの可視性をオンに切り替えます。
- そうすると、Draw Sensor のオプション ボックスにチェックマークが表示されます。

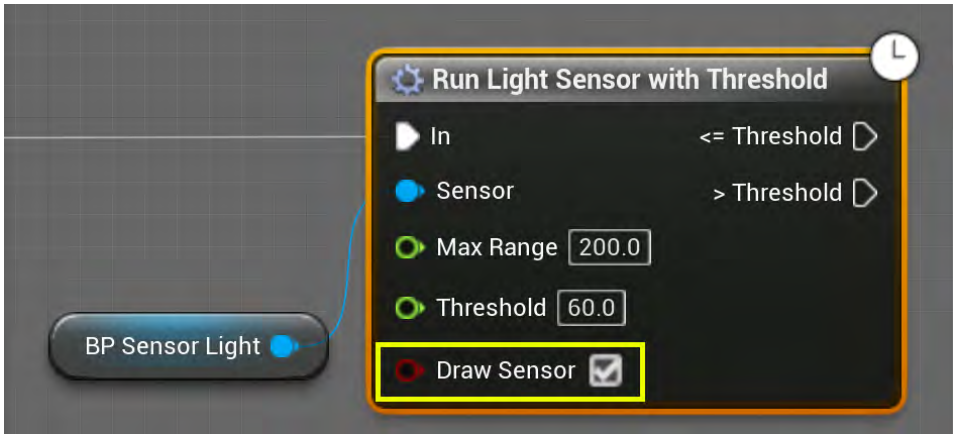


図 3-20

次に、**Run Robot** ノードを Run Motors コメント ボックスからコピーし、その複製をコードの 2 つ目の位置に配置します。

- 灰色の Run Motors コメント ボックスで、**Run Robot** ノードをクリック (選択) します。
- **Ctrl + C キー**を押してコピーします。
- **Ctrl + V キー**を押して貼り付けます (そうすると、元のノードの下にコピーが配置されます)。
- 複製した新しいノードを **Run Light Sensor with Threshold** ノードの右側にドラッグします。
- 次に、**<= Threshold** ピンから新しい **Run Robot** ノードの入力ピンにワイヤーをドラッグして、それらをつなぎます
- そして、**Run Light Sensor with Threshold** ノードの **> Threshold** ピンから、新しい **Run Robot** ノードの入力ピンにワイヤーをドラッグして、両方のしきい値を Run Robot ノードの同じ入力ピンにつなぎます。
- このコードでは、**Light Sensor with Threshold** の実行後に **Run Robot** ノードを呼び出して、ループを作り出します。

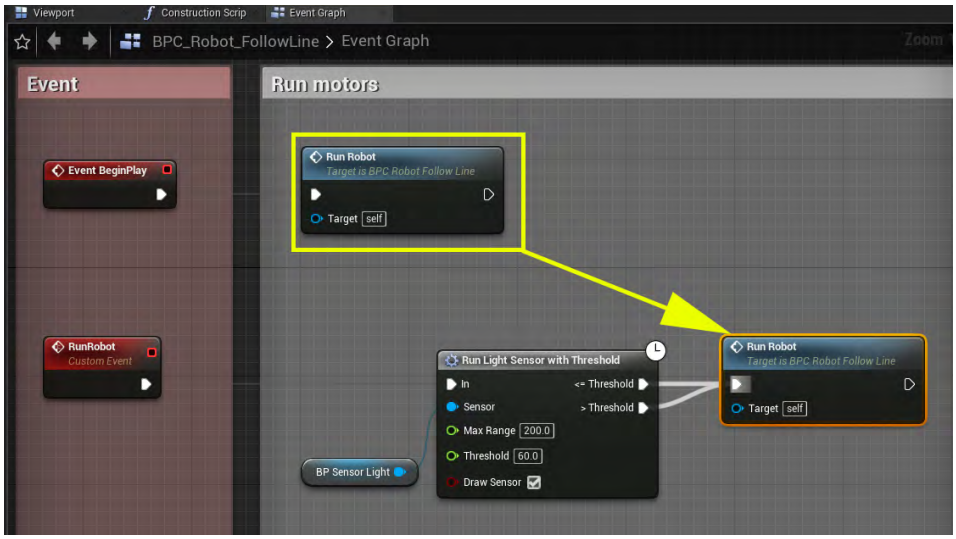
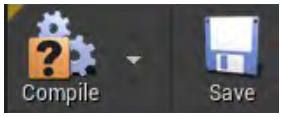


図 3-21

- コードは次のようになっているはずです。
- これで、ゲームをプレイすると、強度の値が変化するのを確認できるようになりました。



- コードを追加したときは毎回「コンパイル」して「保存」することを忘れないでください!

ゲームをプレイして、どのように動作するかを確認しましょう。

- **Map_3-1_FollowLine** タブをクリックします。

白い枠線の値を取得する

- **ビューポート**で **Move Gizmo** を使用して、センサーが黒いラインの左側にあって経路上に留まるように、ロボットを手動で左側または右側に動かします。

図 3-22

- **[Play]** をクリックするか **Alt + P キー**を押します。
- センサーがラインのどこを向いているか、およびロボットの上にある「Light Sensor Feedback」メーターに表示される数値を観察します。

Light Sensor Feedback メーターはロボットの上にバーとして表示されます。この強度メーターは紫色のチェッカー バーとして表示され、黒色 (0) から灰色そして白色 (100) までの階調の陰影で覆われる紫色のチェッカー バーとして表示され、反射光が見えているときに紫色のチェッカー バーが覆われます。

1. 見えている光が少なく、小さい値を返していれば、紫色のチェッカー バーが多く示されます。
2. 見えている光が多く、大きい値を返していれば、灰色から白色への階調が多く示されます。

- センサーのフィードバック値をログ記録し、センサーがどこを見ているかをメモします。
- [Stop (停止)] ボタンをクリックするか Esc キーを押して、プロジェクトのプレイを停止します。
- このプロセスを繰り返して、次の値をログ記録します。
    - センサーが黒いラインを見ているときの値
    - センサーが白い枠線を見ているときの値

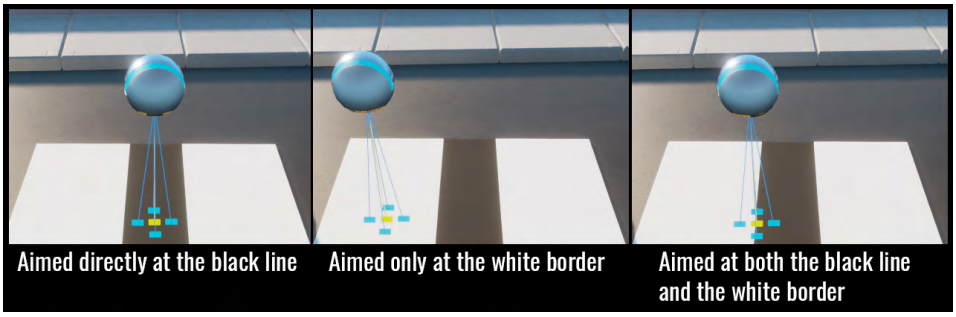
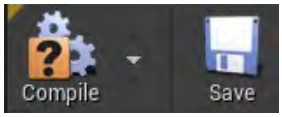


図 3-23

- センサーが黒いラインと白い枠線の両方を見ているときの値



- ブループリントを**コンパイルして保存**します。

生徒および講師への注記: 白い枠線と黒いラインに対してロボットから返される値に大きな差異が得られない場合は、センサーの位置を高くしすぎている可能性があります。ロボットのブループリントのビューポート内でセンサーの配置を調整してから、もう一度試します。

または、光センサーが最適な位置にすでに配置されている**別のロボットを開き**、そのロボットの**イベント グラフ**内でコードを構築し直すこともできます。そうするには、以下の手順を実行します。

- **コンテンツ ブラウザ** パネルで、**「Robots」**フォルダ内の**「L3_Robots」**フォルダを開きます。
- **「BPC_Robot_FollowLine」**という名前のロボットを選択します。
- そして、それを**コンテンツ ブラウザ**からレベルに取り込みます。
- **[Actor Placement Warning (アクタ配置警告)]** が表示されたら、[OK] をクリックします。この警告はこのレッスンには影響を及ぼしません。

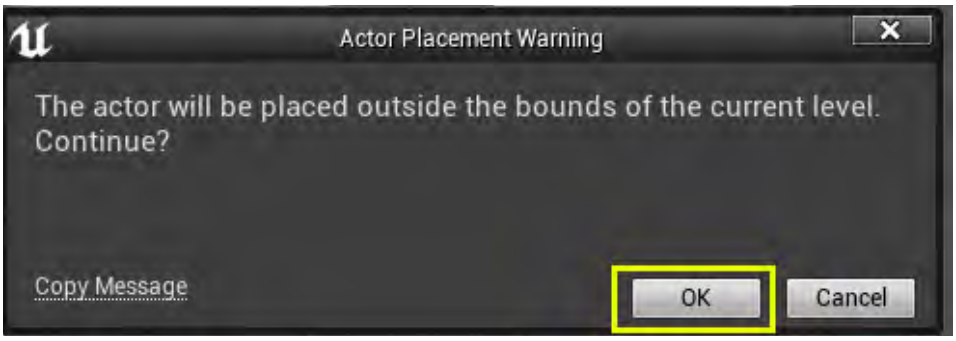


図 3-24

- **ビューポート**でこのロボットを選択します。
- **[Details] パネルで [Edit Blueprint] をクリック**し、**[Open Blueprint Editor]** をクリックします (**BPC_Robot_FollowLine** の新しいアクティブな Unreal タブ内にイベント グラフが表示されているはずです)
- 実習 2 の図 3-17 に戻って、このロボット用のコードの構築を進めます。

実習 3:必要なしきい値を決める

しきい値とは

このプロジェクトでは、センサー返す値は 0 ～ 100 の範囲です。ロボットは走行中はいつでもセンサーからのフィードバックに基づいて 2 つの異なる行動のどちらかに決める必要があります。
しきい値は、行うアクションを選択するようにロボットの条件文に指示する、単一の数値です。条件式では、しきい値と等しいかそれより大きい (>=) 値をセンサーが見ているかどうかが問われ、そうであればあるアクションをトリガーし、しきい値より小さい (<) 値をセンサーが見ていれば、別のアクションをトリガーします。つまり、しきい値は、ロボットが白い枠線と黒い経路ラインのどちらの上にいるかをロボットが判断するために使用される「境界」です。

次の質問への回答を見つけ出す必要があります。

- 光センサーをどのように使用して、ロボットが「見ている」ものからフィードバックを取得するか
- ラインが検出されると、どのような値が示されるか
- ラインが検出されないと、どのような値が示されるか

注記:Notepad または学校やクラス用の共有ドキュメント システムを使用して、以下の手順を進めるときに光センサーの値をログ記録します。

ここで、経路の黒い部分を一貫して見るために必要となる、光センサーのしきい値を定める必要があります。黒い経路ラインと白い枠線 (地面) の値をメモしているので、センサーからの入力の範囲全体を、「地面が見えている」と「ラインが見えている」に分けるしきい値を定めます。

- 白い枠線 (地面) の値と配送経路ラインの値をログ記録した自分のメモを参照します。
- 次の式に基づいてしきい値を決めます。**((最高値 – 最低値) / 2) + 最低値 = しきい値**
    - たとえば、最高値 (白い枠線) が 90 で、最低値 (黒い経路ライン) が 10 であれば、この式は **((90 - 10) / 2) + 10 = 50** になります。
- しきい値を観察してログ記録します。

このプロジェクトには、しきい値計算機能を組み込んでいて、インターフェースの左上にある [Window (ウィンドウ)] メニューからアクセスできます。

[Window] > [Editor Utility Widgets (エディタ ユーティリティ ウィジェット)] > [Threshold_Calculator]

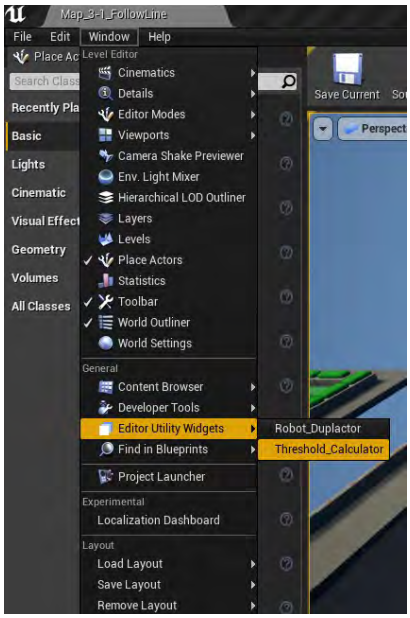


図 3-25

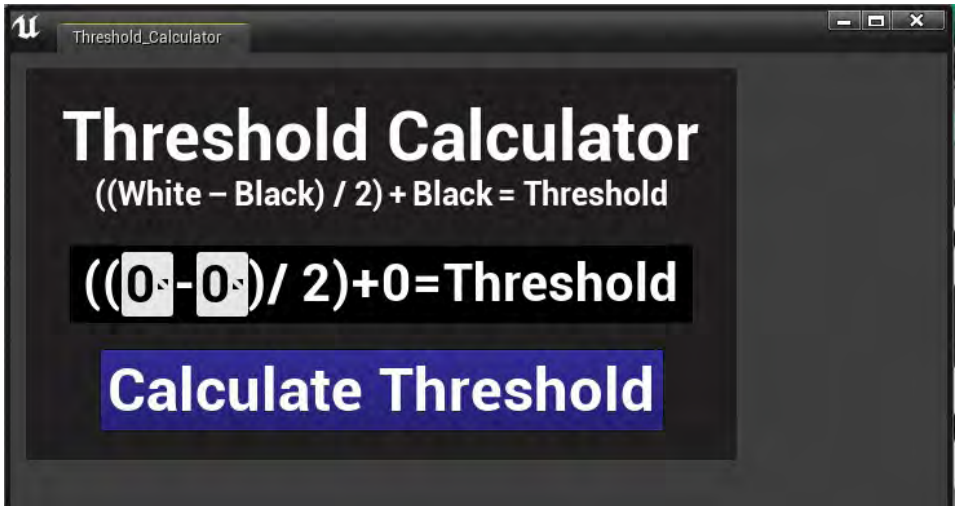


図 3-26

実習 4:ラインを見つける

光センサーからの入力をテストし、現在のフィードバックに基づいて判断します。そして、そのフィードバックに基づいて、モーターを回転させるように指示します。いくつかのモーター コマンドを追加して、ロボットが独力で走行および旋回を行うようにする必要があります。

ロボットを配送経路ラインに合わせる

ロボットを動かす前に、ラインを基準としてロボットが向いている方向に注目します。その方向に応じて、動きをコーディングするときに行う選択を決定します。実習のこの部分では、黒いラインの左側にある白い枠線上に光センサーがある場合に、ロボットを経路の方に向けます。

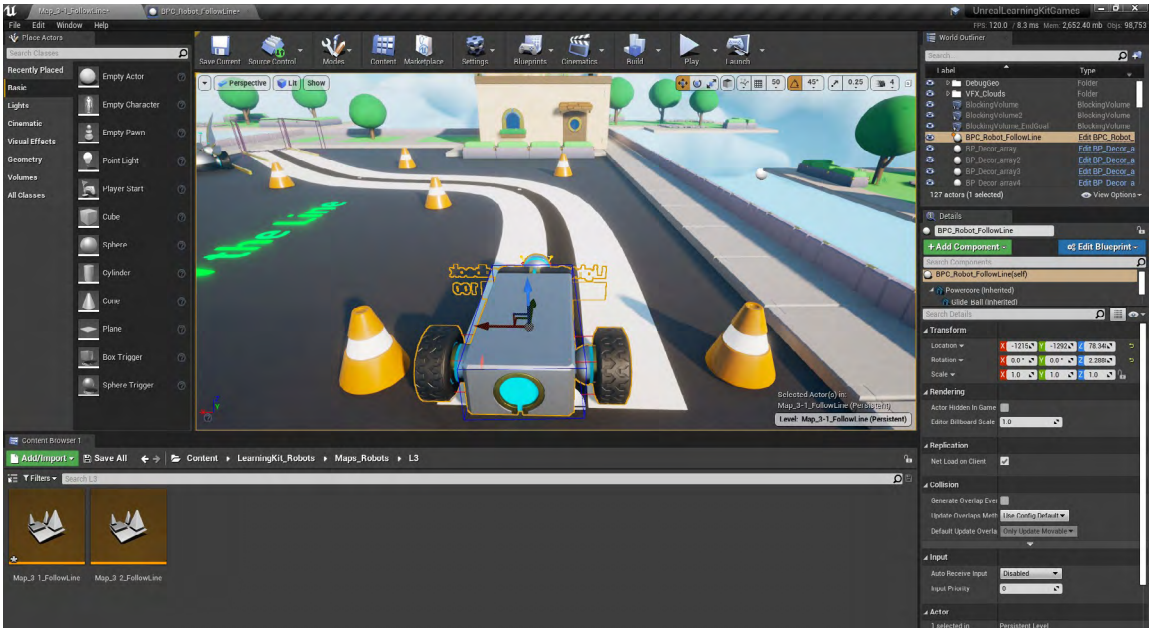


図 3-27

センサーからの入力に対応する

ロボットがセンサーからの入力値を読み取ることができる場合、センサーからのフィードバックに基づいて特定の行動を取ることができます。まず、ラインが見えていれば、そのラインから離れるようにアーク旋回するようにします。

アクティビティ:ここでは、黒いラインが見えている (または見えていない) ときにフィードバックを与えるように、しきい値を設定します。入力値がしきい値を上回っていれば、白い枠線 (地面) が見えています。

ブループリント エディタでロボットを開きます。	
ブループリント エディタをすでに開いている場合	**ブループリント エディタを開いていない場合**
次のラベルが付いているタブに切り替えます。**BPC_Robot_FollowLine**	• ロボットを選択します (右側にある [Details] パネルで **BPC_Robot_FollowLine** がアクティブなアクタとして表示されるはずです)。 • **[Details] パネル**で **[Edit Blueprint]** をクリックし、**[Open Blueprint Editor]** をクリックします。

- ブループリントのビューポートを表示していれば、**イベント グラフ**をクリックします。
- **BPC_Robot_FollowLine** のブループリント エディタ内に**イベント グラフ**が表示されているはずです。

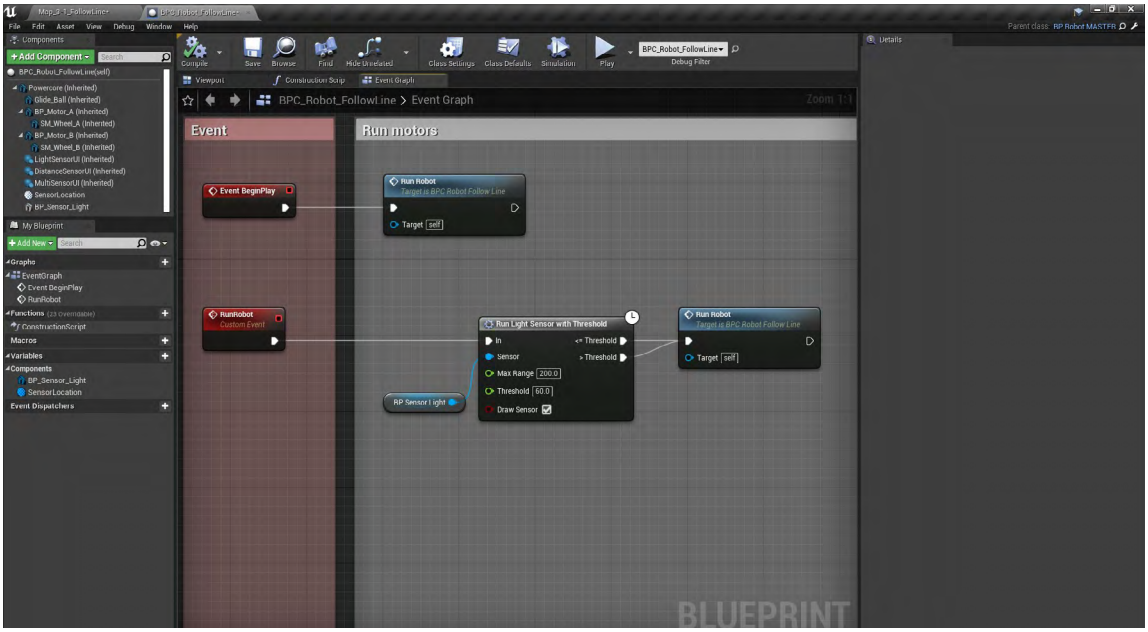


図 3-28

以前のコードを編集して新しいコマンド用に空きスペースを作る

Run Light Sensor with Threshold ノードと最後にある **Run Robot** の間に新しいコマンドを配置できるように、コードに空きスペースを作る必要があります。

- Run Robot ノードを選択してもっと右側に動かします (図 3-29 を参照)。

Run Light Sensor with Threshold ノードとコードの最後にある **Run Robot** の間のワイヤーを切断する必要があります。新しい複数のノードを追加し、それらをそれぞれ異なるコマンドにつなぎます。

- **Run Robot** の入力ピンつながれているワイヤーを切断するには、**Alt キー**を押したままにして、そのワイヤーまたは **Run Robot** ノードの白色の実行ピンをマウスで左クリックします。

ブループリントは次のようになっているはずです。

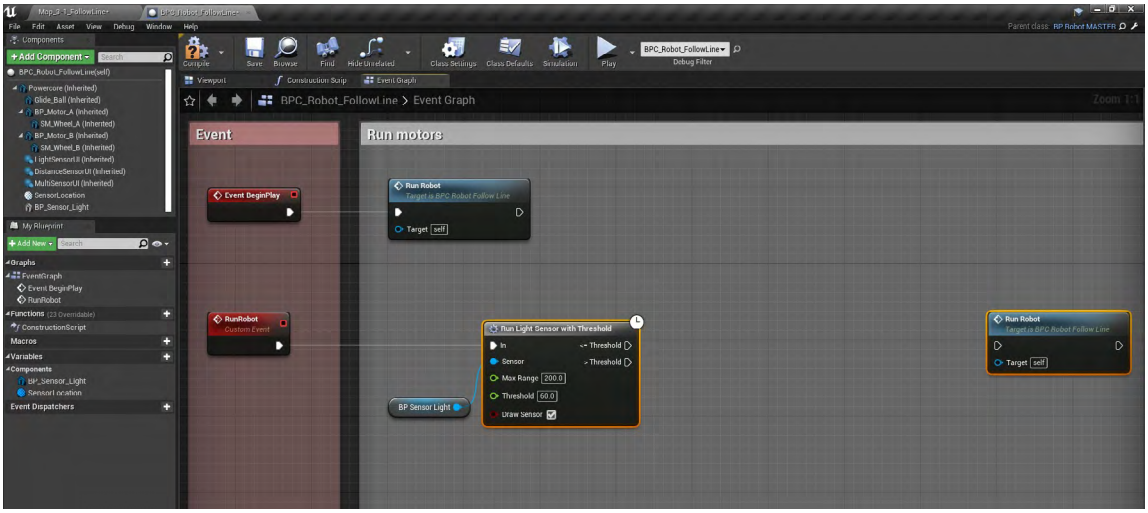


図 3-29

モーターのコマンド

- **Run Motor** ノードは、自身につながれているモーターを制御します。図 3-30 は、**BP_Motor_A** コンポーネントを旋回させるためのブループリントの設定を示しています。**Speed** 入力は、モーターを回転させる速度を制御し、正の数値であればモーターをある方向に回転させ、負の数値であればモーターを逆方向に回転させます。このフィールドは -100 ～ 100 の範囲の値を取ります。

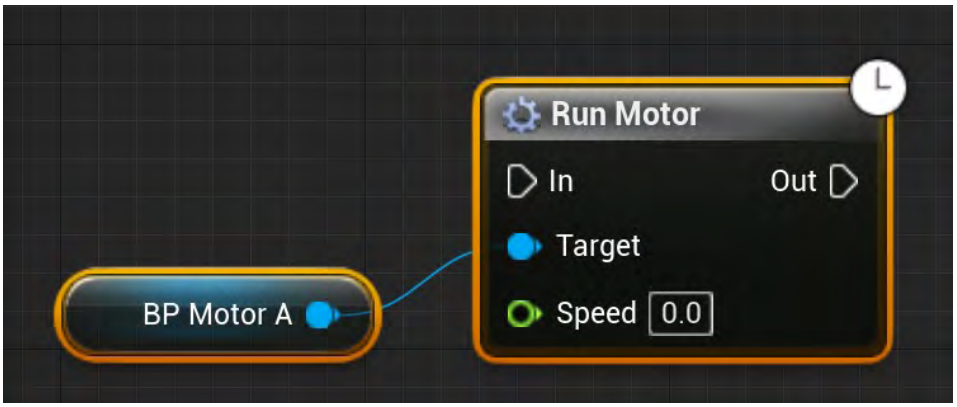


図 3-30

- **Stop Motor** ノードは、自身につながれているモーターを制御します。図 3-31 は、**BP_Motor_A** が回転していれば停止するためのブループリントの設定を示しています。**Coast** 入力は、オンであれば惰力で回転してからゆっくりと停止すし、オフであれば直ちに回転を停止するように、モーターに指示します。図 3-31 では、オフになっているのでモーターは直ちに停止します。

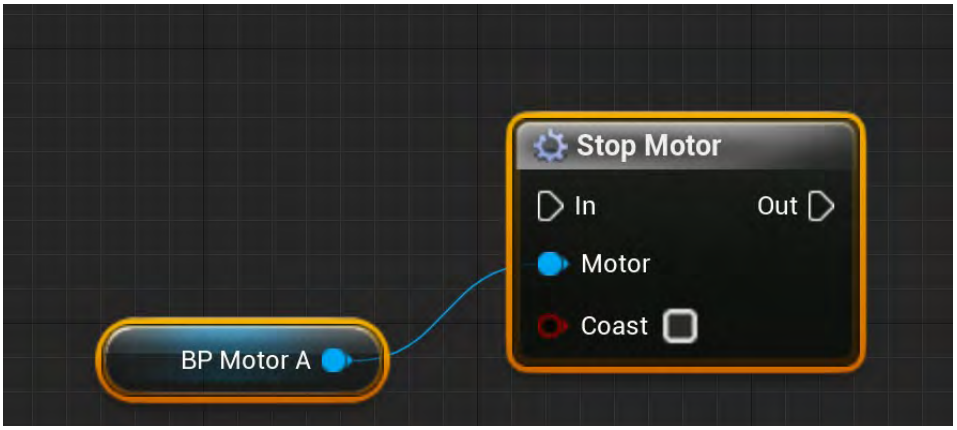


図 3-31

白い枠線が見えているときのアクション

入力値がしきい値を上回っていれば、白い枠線 (地面) を見ています。ロボットが黒いラインを絶えず探すようにしたいので、ロボットが白い枠線の上にあるか、または黒いライン以外のどこかにあれば、黒いラインの方に向きを変えるようにします。

黒いラインの方に向きを変えるようにロボットをコーディングする

ロボットは現在は、黒いラインの少し左側で、白い枠線の上に配置されています。
ロボットを右向きに (黒いラインの方に) 旋回させたいので、左側のモーター (モーター A) を前進させ、右側のモーターを静止させる必要があります。

まず、**BP_Motor_A** ノードを追加しましょう。
[Components] パネルで、**BP_Motor_A** を**クリック**し、それをイベント グラフに**取り込みます**。

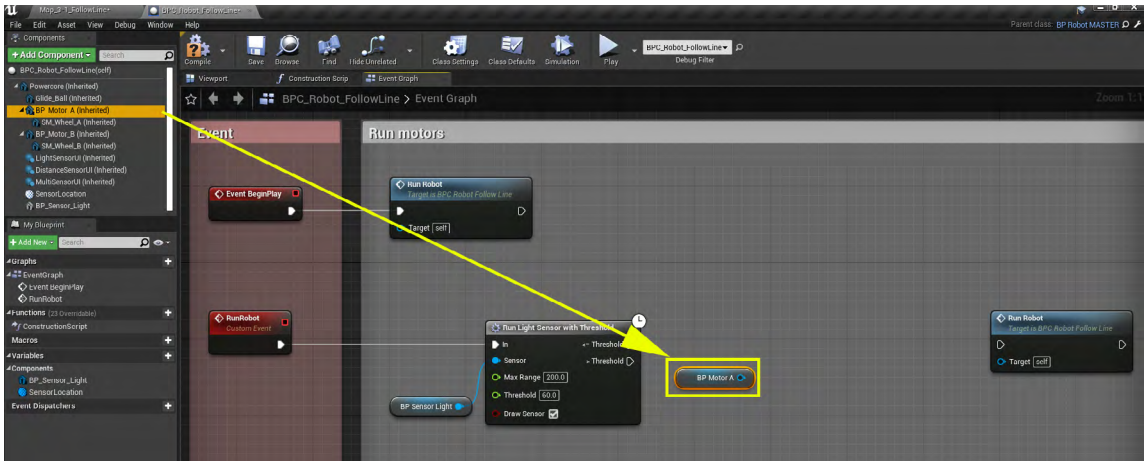


図 3-32

次に、**BP_Motor_A** の青色の出力ピンを**クリック**して右側に引き出します。

- 「run motor」と入力して検索し、**Run Motor** ノードを選択します。

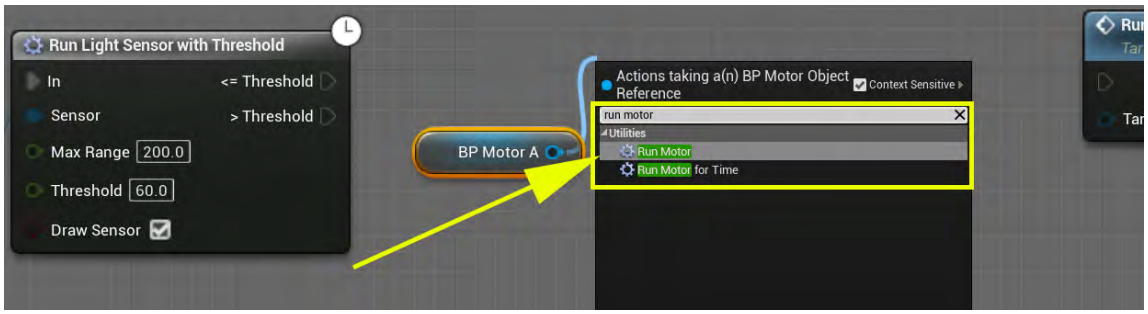


図 3-33

- ロボットが黒いライン上にいないときの方が多くの反射光が見えているので、光センサーからの入力値はしきい値よりも大きいことを思い出してください。
- **Run light sensor with Threshold** ノードの白色の **> Threshold** ピンを **Run Motor** ノードの入力ピンにつなぎます (クリックしてドラッグ)。

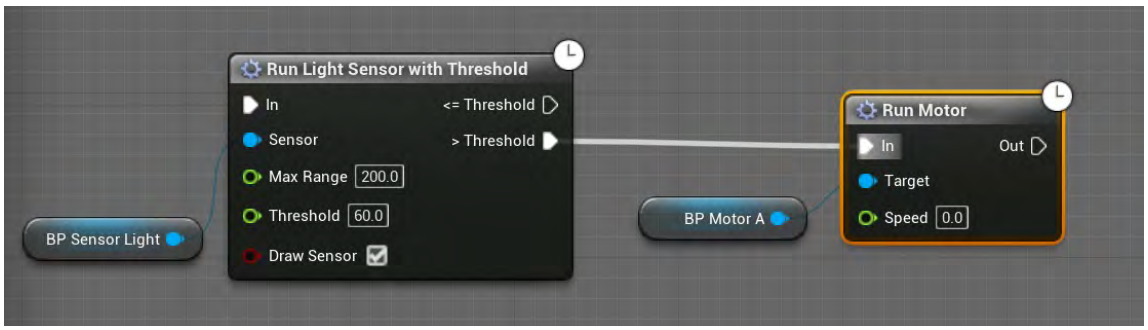


図 3-34

- **Run Motor** ノードの Speed を、0 ～ 100 の範囲の値 (30 を推奨) に設定して、ロボットを前方に押し出す方向にモーターを回転させます。そうすると、ロボットは黒いラインの方に向きを変えます。
- **[Components] パネル**で、**BP_Motor_B** をクリックして **Run Motor** ノードの右側に引き出して、図 3-35 に示しているように、コードのフロー内に配置します。

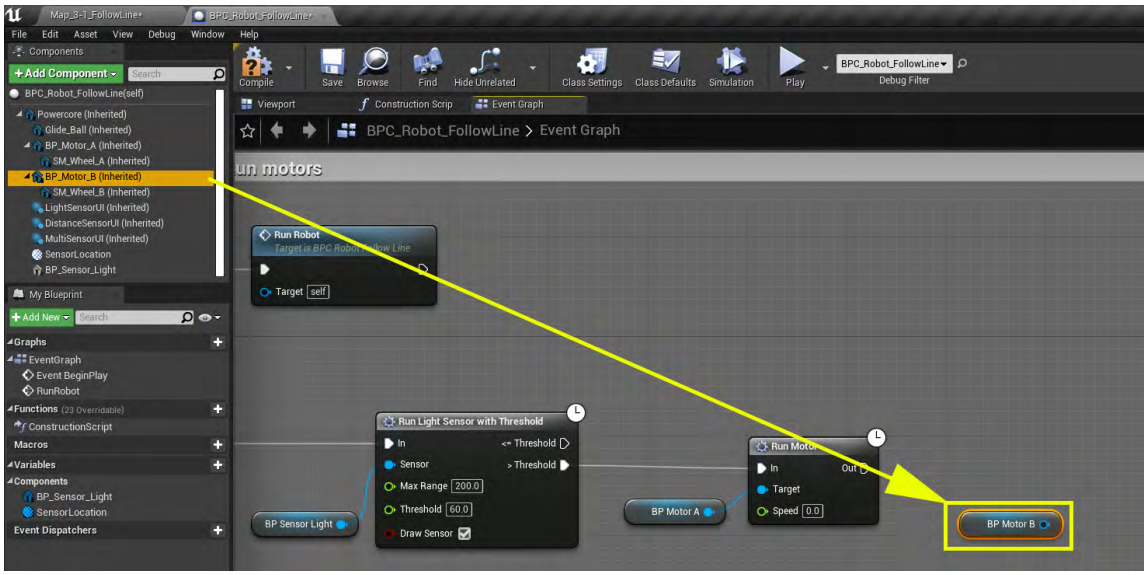


図 3-35

- **BP_Motor_B** の青色の出力ピンを**クリック**して右側に引き出します。
- 「stop motor」と入力して検索し、Stop Motor ノードを選択します。

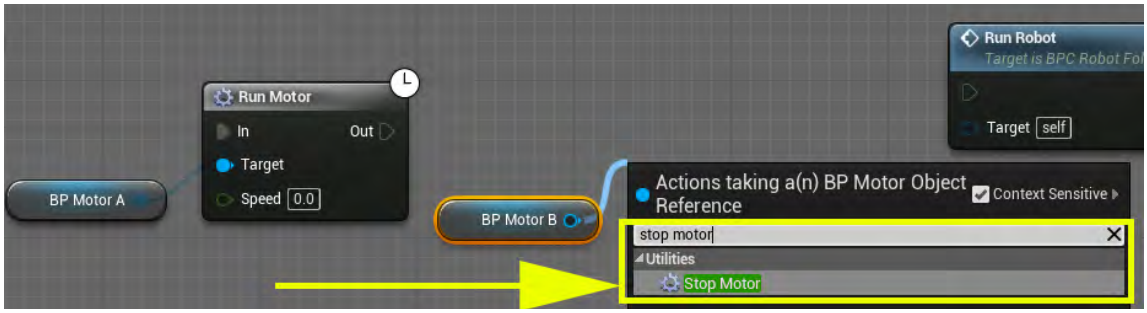


図 3-36

Run Motor ノードを Stop Motor ノードにつなぎます。

- **Run Motor** の出力ピンを**クリック**して (白色の) ワイヤーを引き出し、**Stop Motor** の入力ピンにつなぎます。

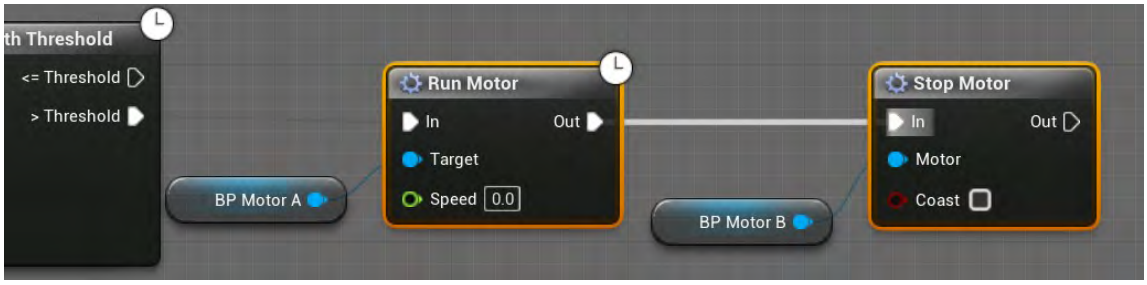


図 3-37

Stop Motor ノードを、右側にあるコピーして作成した **Run Robot** ノードにつなぎます。

- **Stop Motor** の出力ピンを**クリック**して (白色の) ワイヤーを引き出し、**Run Robot** ノードの入力ピンにつなぎます。

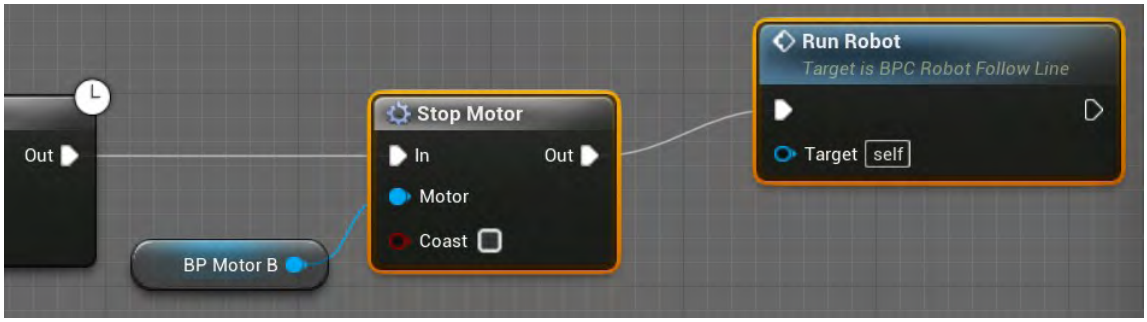


図 3-38

新しい 4 つのコマンドを 1 つのグループにするコメント ボックスを作成します。

- 投げ縄ツールやマーキー ツールのように、クリックして 4 つのノードすべてを囲むようにドラッグして、それらすべてを一度に選択します。
- **C キー**を押して、選択したノードを囲むコメント ボックスを作成します。
- 「Turn toward line」と入力して Enter キーを押します。
- そうすると、コードのサブセクションに名前が付きます。

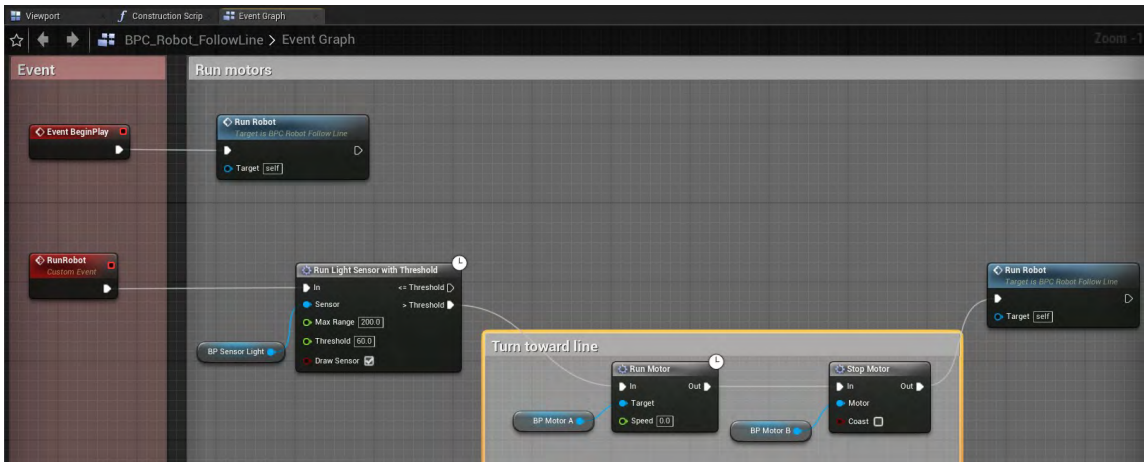
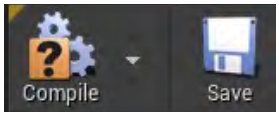


図 3-39

- **コンパイル** して**保存**します。



- **Map_3-1_FollowLine** タブをクリックしてマップに戻ります。
- 経路の開始位置の、黒いラインの左側の白い枠線の上にロボットを配置します。
- **1** キーを押すと、ズームアウトされてマップ全体が表示されます。
- ゲームをプレイする

パフォーマンスに関する注意事項

これらのロボットのパフォーマンスは、作業に使用しているコンピュータの性能に直結しています。**Ctrl + Shift + H** キーを押すことで、プロジェクトが実行されている 1 秒あたりのフレーム数 (fps) が示される表示モードを切り替えることができます。これらのレッスンの開発は、120 fps を達成できるコンピュータ上で行われました。ロボットが期待どおりに動作しない場合、使用しているコンピュータの性能が原因である可能性があります。気を落とすことはありません。使用しているコンピュータ上で実行している他のアプリケーションを終了すると、パフォーマンスが向上する可能性があり、いつでもやり直して、もっと多くのデータを収集し続けることができます。

- 観察、テスト、デバッグを行って、白い枠線が見えているか、または黒いライン以外のどこかが見えているときに、ロボットが黒いラインの方に向きを変えることを確認します。

実習 5：分岐を完成させる

ロボットの行動を観察するときに、連続して黒いラインの方に向きを変えていて、黒いラインが見えているときに何をすべきかまだわかっていないことに注意してください。ここでは、黒いラインが見えているときにしきい値と等しいかそれより小さい (<=)
センサーからの入力を見つけるようにしたいので、分岐の反対側を
作成します。

経路ライン (黒色) が見えているときのアクション

次に、ロボットが黒い経路ラインを見ているかどうかをテストするためのノードを追加し、センサーの出力を新しい分岐につなぎ、モーター関数を追加し、モーターの速度を変更して正しい方向に旋回します。

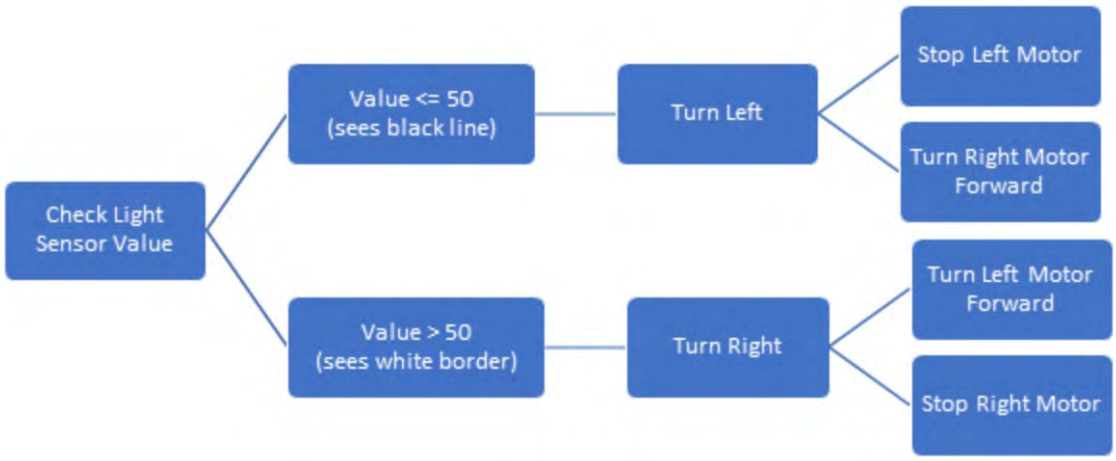


図 3-40

上記のロジック図を復習します。センサーがしきい値 (50) 以下である値を返せば、センサーは黒いラインを見ています。センサーが黒いラインを見ているときに、何が起きてほしいですか。ラインに沿って前進してほしいと思うかもしれませんが、黒いラインから離れるように動いてから黒いラインの方に向きを変えることで、ロボットが途中で「小刻みに動く」ようにします。そうすることで、ラインをガイドとして利用して連続的に前進します。そうするために、黒いラインが見えているときに、ロボットが左向きに旋回する、つまり黒いラインから離れるようにします。

1 つ目の分岐から現在のコードのサブセクションを複製することで、すべてのノードが含まれているもう一方の分岐を簡単に作成し、それを編集して、センサーがしきい値以下の値を返したときに逆方向に動くようにします。

ブループリント エディタを開きます。

- **ビューポート**にあるロボットを**クリック**します (右側にある [Details] パネルで **BPC_Robot_FollowLine** がアクティブなアクタとして表示されるはずです)。
- **[Details] パネル**で **[Edit Blueprint] をクリック**し、**[Open Blueprint Editor]** をクリックします。
- ビューポートを表示していれば、**イベント グラフ**をクリックします。
- **BPC_Robot_FollowLine** のブループリント内に**イベント グラフ**が表示されているはずです。

前回コメント ボックスにしたコードをコピーして貼り付けることで複製します。

- **Turn toward line** コメント ボックス全体を選択します (クリックして「Turn toward line」サブセクション全体を囲むようにドラッグし、5 つの要素 (4 つのノードとコメント ボックス) すべてがオレンジ色の枠で囲まれて、選択されていることが示されている)。
- ***ヒント：Shift キーを押したままにしてクリック**することで、選択されているノードに追加して、すべてのノードが選択されている状態に*
  することもできます。
- 選択されているノードを**右クリック**し、ポップアップ ダイアログ ボックスで **[Copy (コピー)]** を選択します (または **Ctrl + C** キーを押します)。
- マウス カーソルを、ブループリントの灰色の空きスペースの上部に移動させます。
- **Ctrl + V** キーを押して、複製したサブセクションを**貼り付けます**。そうすると、元のコードの上方に複製が貼り付けられます。
- 必要に応じて、複製されたノードをクリックしてドラッグし、元のコード ボックスに揃えます (図 3-41 を参照)。

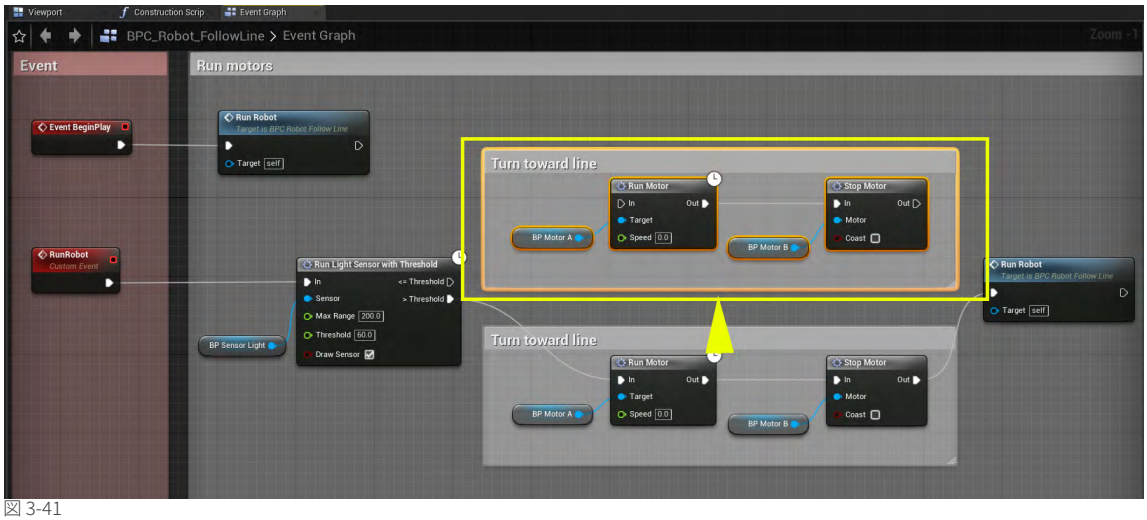


図 3-41

新しいコメント ボックスの名前を変更します。

- ブループリントの灰色の領域のどこかをクリックして、新しいセクションの選択を解除します。
- 上にある新しいボックスを**右クリック**して選択します。
- ダイアログ ボックスで **[Rename (名前変更)]** を選択します。
- 「Turn away from line」と入力して **Enter** キーを押します。
- そうすると、コード ボックスの名前が変更されて、2 つのボックスがそれぞれ固有の名前を持つようになります (上にあるボックスが「Turn away from line」で、下にあるボックスが「Turn toward line」です)。

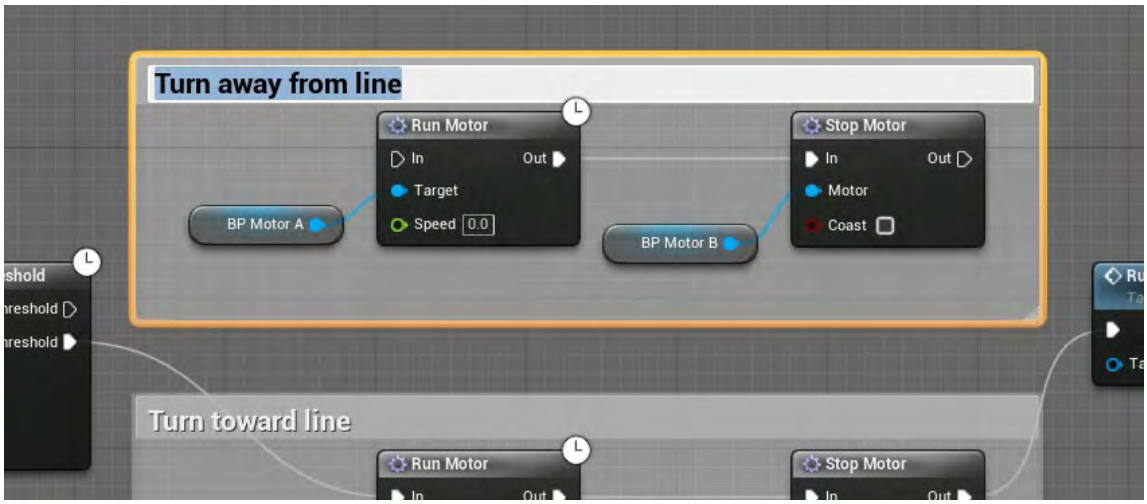


図 3-42

ロボットが旋回する向きを逆にしてラインから離れるようにすることを素早く簡単に行うために、**BP_Motor_A** ノードと **BP_Motor_B** ノードを入れ替えます。

- **Alt** キーを押したままにして、「Turn Away From Line」ボックス内のモーターにつながっている青色のワイヤーを
  クリックします。
- **BP_Motor_B** をクリックしてドラッグし、**Run Motor** ノードの左側に移動させます。
- **BP_Motor_A** をクリックしてドラッグし、**Run Motor** ノードの右側に移動させます。
- コードは図 3-43 のようになっているはずです。

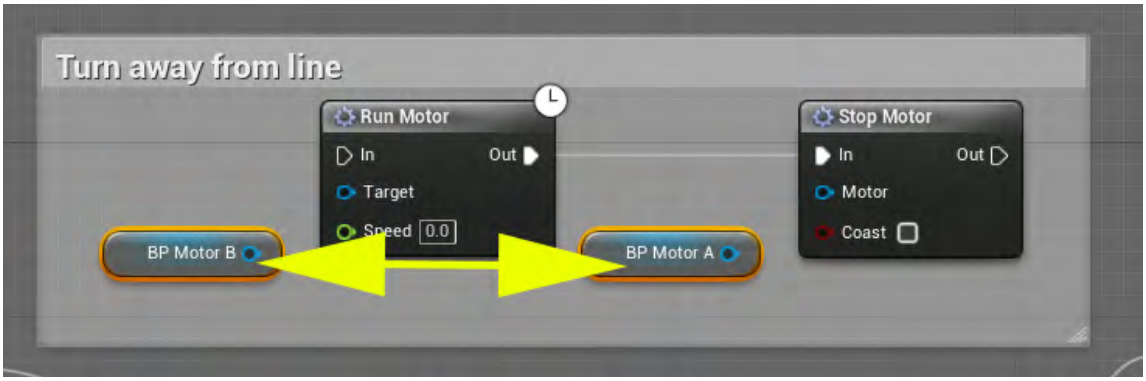


図 3-43

ノードをつなぎ直す

- **BP_Motor_B** のピンをクリックして (青色の) ワイヤーをドラッグし、**Run Motor** ノードの Target ピンにつなぎます。
- **BP_Motor_A** のピンをクリックして (青色の) ワイヤーをドラッグし、**Stop Motor** ノードの Motor ピンにつなぎます (図 3-44 を参照)。

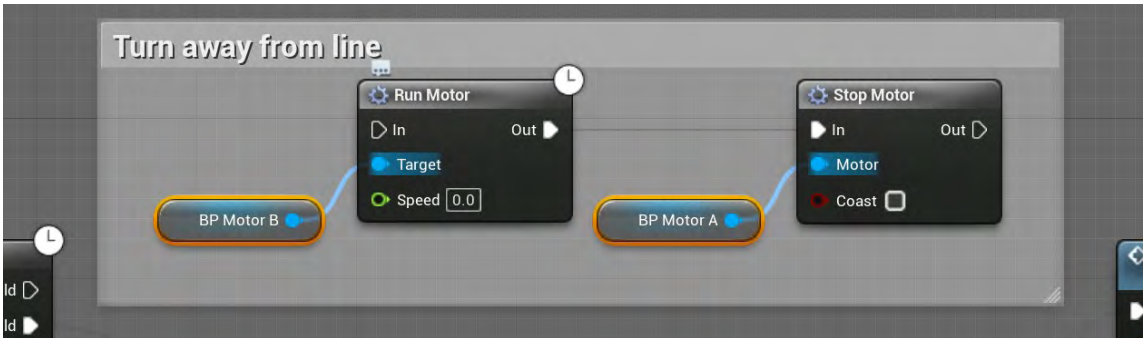


図 3-44

- **BP_Motor_B** の **Speed** のテキスト ボックス内でクリックして好みの数値を入力することで、速度を -100 ～ 0 の範囲の値に設定します。(30 に設定することをお勧めします。この値は、逆向きであること以外は他の **Run Motor** ノードと同じにすべきです)。
- **モーター B** は反転して配置されているので、前進するには負の値を指定する必要があることを思い出してください。

すべてのコードをつなぐ

- **Run Light Sensor with Threshold** ノードの **<= Threshold** ピンを**クリック**して (白色の) ワイヤーを引き出し、**Run Motor** ノードの入力ピンにつなぎます。
- **Stop Motor** ノードの出力ピンを**クリック**して (白色の) ワイヤーを引き出し、**Run Robot** ノードの入力ピンに
つなぎます。

コードが図 3-45 のようになっていることを確認します。

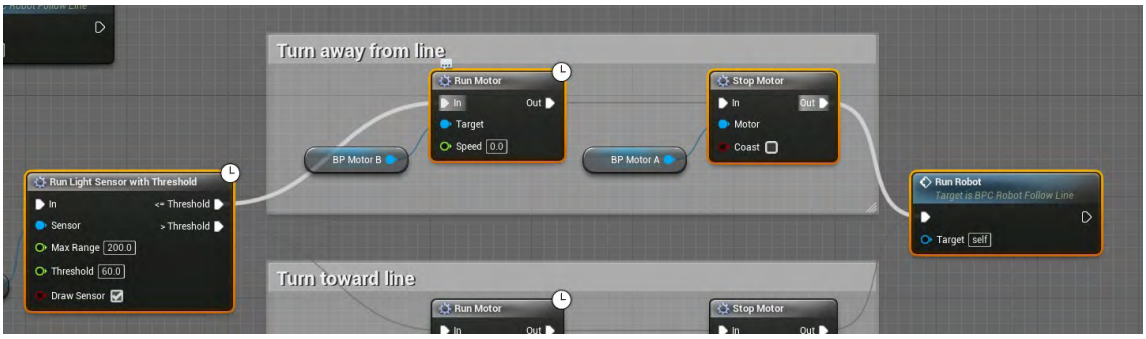
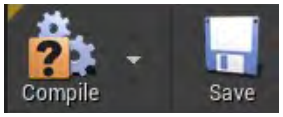


図 3-45

- **コンパイルして保存**します。



- コードは図 3-46 のようになっているはずです。ロボットをテストするときに、Speed の値をあれこれ試すことや、**Coast** の **オン/オフ**を試してみることができます。

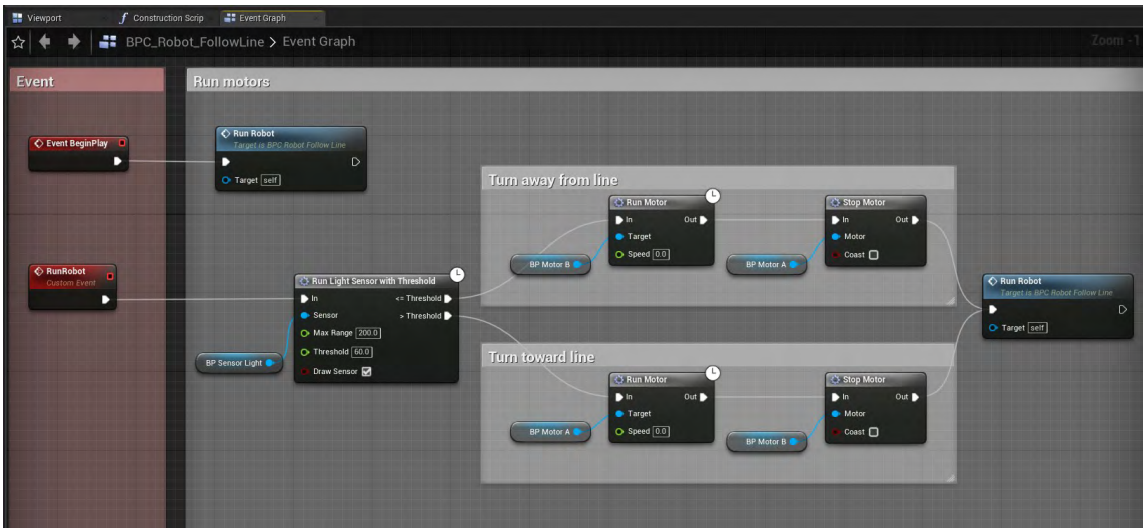
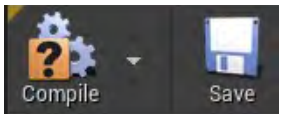


図 3-46

- コードを追加したときは毎回「コンパイル」して「保存」することを忘れないでください!



- **Map_3-1_FollowLine** タブをクリックしてマップに戻ります。
- 経路の開始位置の、黒いラインの左側の白い枠線の上にロボットを配置します。
- **1** キーを押すと、ズームアウトされてマップ全体が表示されます。
- ゲームをプレイする
- Speed の値をあれこれ変更して試し、コードをテストして、ロボットが正確に正しい行動をどの程度取るかを確認します。

結果を観察する

ロボットを数回試して、プレイごとのロボットの開始位置をメモし、開始位置を基準としたロボットの行動をメモします。ロボットの行動は多少変化するので、必ず 2 回以上試してください。

課題

ロボットが経路の終点までずっと経路ラインに追従できるようになるまで、ロボットでのセンサーの位置、ロボットの開始位置、およびコードを微調整し続けます。

- ロボットが黒いラインを見るとそのラインから離れるように動くことに成功するまで、コードを編集してトラブルシューティングを行います。次に質問を活用してデバッグを進めます。
    - ロボットがラインに近づいたときにどうなりましたか。ライン上に留まっていましたか。
- ロボットがライン上にいるときにラインから離れるように向きを変えるようになるまで、修正を続けます
    - ロボットがラインを離れたときにどうなりましたか。ラインをもう一度見つけるすることができましたか。
- ロボットがライン上にいないときにラインの方に向きを変えるようになるまで、修正を続けます
    - ロボットの動きが速すぎて、経路を見失いましたか。ロボットは経路から離れて旋回していますか。
- ロボットはラインに沿って前進しましたか。
- 最も信頼性の高い結果が得られる値を見つけ出します

ヒント:

- 問題が発生したら、ロボットがより正確に旋回できるように速度を下げて試してみます。
- ロボットの開始位置を左側から右側に変えたり、経路からの光センサーの高さを変えたりして、あれこれ試してみます。

注記: ロボットは目的地で停止します。このアクティビティの現行のロボット モデルは、目に見えない壁であるブロッキング ボリューム ボックスにぶつかると停止するので、この行動をコーディングする必要はありません。

拡張アクティビティ

複数の光センサー

ライン追従行動をもっと滑らかな動きにする (何度も「小刻みに」後退したり前進したりしない) にはどうすればよいでしょうか。センサーを 2 つ使用して動きを滑らかにすることを検討します。

2 つ目のセンサーを使用する場合、センサーはそれぞれどこに配置すべきでしょうか。

- 1 つ目の光センサーを移動させる
- 2 つ目の光センサーを配置する
- センサーを追加または移動した場合は、必ずしきい値を算出し直す

2 つ目の光センサーを使用する場合、それぞれのセンサーへの指示に対してコードはどのように変わりますか。

- 既存のコードを再利用できるか。
- 3 つの状態を持つことができるか。たとえば、左側のセンサーでラインが見えたら左向きに旋回する、右側のセンサーでラインが見えたら右向きに旋回する、ラインが見えなかったら直進する、といったことです。

2 つのセンサーを使用する場合のセンサーの位置：

図 3-47

速度

- 達成する速度はユーザーに任されています。
- どの程度速くできますか。
- 速度を上げるとどのような問題が生じますか。

新しいコーディング手法

- 2 つのセンサーを同時に実行するため、つまり両方のセンサーを一度に 1 つずつではなく両方をほぼ同時に実行させるために、シーケンス コマンドを使用することを学びました。

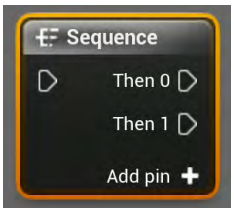


図 3-48

- 光センサーのしきい値テストを使用した条件文。

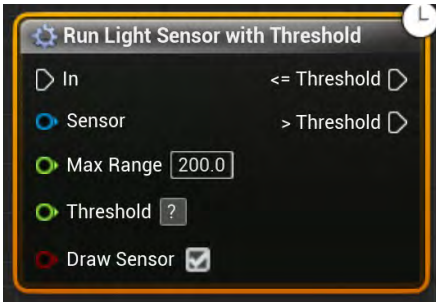


図 3-49

- ループの設定：どこで開始するか、どこで終了するか、無制限の反復を実行する方法

復習

- センサーの必要性、およびセンサーが入力をどのように解析しているかを理解する。
- ロボットのさまざまな位置に配置した場合のセンサーの有効性を理解する。
- センサー値の範囲をブール値に変えるしきい値の活用法を理解する。センサーがブール値を使用して分岐に沿って実行する。
- センサーがロボットの動きを決定するのに役立つコマンドをコーディングを理解する。
- コーディングのつながり：しきい値テストを使用した条件文。ループを使用してアクションを反復する。
- 課題、成功したこと、理解したことのメモおよびデモ

リソース

- 観察ログ – 空白のログと、拡張アクティビティ用のログ

評価

基準

概念	抜群	熟練	適格	未熟
ロボット設計の概念	ロボットのセンサー コンポーネントとそれがどのように機能するかを他の人に説明できる。プロジェクトで、レッスンでのセンサーの使用およびロボットでの最適な位置を示すことができる。デモまたは生徒のドキュメントで、コンポーネントの定義が示されている。	ロボットでのセンサーの使用が示されている。ロボットのコンポーネントについて完全に理解していることが、講師に示されている。	ロボットをセンサーに対応させることはできるが、センサーについて意義のある知識を提供していない。ハードウェア コンポーネントについての基本的な知識を持っていることが示されている。	ロボットのセンサーとその機能を理解している根拠がない。ハードウェア コンポーネントについての知識を持っていることが示されていない。
ソフトウェアの概念	コマンドの複雑な組み合わせとプロジェクトの目標が示されている。センサーの使用、センサーの位置、および結果としてのロボットの望ましい位置や行動に必要なコマンド グループについて説明できる。複数のコマンドまたはコマンド グループがプレゼンテーションまたはドキュメントで示されている。	センサーの使用、センサーの位置、および結果としてのロボットの望ましい位置や行動に必要なコマンド グループについて理解している。複数のコマンドまたはコマンド グループについて生徒のプレゼンテーションで触れられている。	促されれば、センサーの使用、センサーの位置、および結果としてのロボットの望ましい位置や行動に必要なコマンド グループについて基本的な知識を持っているが、デモでは触れていない。	コマンドおよびその機能について理解している根拠がない。生徒のプレゼンテーションでコマンドについて触れていない。
コーディングの概念	コードのエラーをデバッグできる。コードの複雑な組み合わせと独自の使用方法が示されている。ほとんどのセクションのコードを説明できる。生徒のプレゼンテーションまたはドキュメントにコードが記載されている。	デフォルト設定、ループ、条件付きコマンドについての知識が示されている。生徒は自身のプロジェクトで各コマンド タイプのコードをそれぞれ 1 つ以上を参照している。	促されれば、生徒はコードについて基本的な知識を持っているが、生徒のプレゼンテーションでは触れていない。	コマンドのコードおよびその機能について理解している根拠がない。生徒のプレゼンテーションでコードについて触れていない。
実世界の概念	実世界でのロボット、コーディング、移動の利用を生み出す革新的なアイデアを持っている。理解していることを示しており、そのことをドキュメントに記載している。	実世界のアプリケーションでのコーディング、移動、ロボットの利用について理解している。生徒のプロジェクトのプレゼンテーションに 1 つ以上の例が記載されている。	実世界のアプリケーションでのコーディング、移動、ロボットの利用について基本的な知識を持っている。促されれば言葉で説明できるかもしれないが、プレゼンテーションでは触れていない。	コーディング、移動、ロボットの実世界のアプリケーションでの利用について理解している根拠がない。
課題アクティビティ (ライン追従ロボット)	完全自動化されたライン追従ロボットのデモを行い、コーディング プロセスの反復を示し、試行、課題、成功したことについてのドキュメントを提供している。	ライン追従ロボットをコーディングし、構築プロセス中にコーディング、観察、テストを複数回反復した。	ライン追従ロボットの仕組みについて基本を理解している。ライン追従ロボットの一部の要素をコーディングできた。	ライン追従ロボットのデモを行うことができなかった。ライン追従ロボットの仕組みを理解していなかった。

生徒向けガイド

仮想ロボットのトレーニングを学ぼう

レッスン 3：自動運転車