実行環境およびlibbats2.0のインストール

simspark, rcssserver3d, libbats2.0の順にインストール

予め下記のパッケージをインストールしておく

```
g++ subversion cmake libfreetype6-dev libode-dev libsdl-dev ruby1.8 ruby1.8-dev libdevil-dev
 libboost-dev libboost-thread-dev libboost-regex-dev libxml2-dev latex2html imagemagick
 doxygen libsigc++-2.0-dev libeigen2-dev
```

☆ simspark, rcssserver3dのインストール

- svn co https://simspark.svn.sourceforge.net/svnroot/simspark simspark
  - sparkディレクトリまたはrcssserver3dディレクトリに移動
- mkdir build; cd build
- cmake ..
- make
- sudo make install

両方インストールできたら，「rcsoccersim3d」でサーバ＋モニタ起動

☆ libbats2.0のインストール（既に導入済みの方も再インストールをお願いします）

- tar zxvf ~~~.tar.gz
- ./configure
- make
- sudo make install

☆ opuCI_3D_campのコンパイル

- tar zxvf ~~~.tar.gz
- ./configure
- make

コンパイルが終わったら，サーバ起動後に「./start.sh」でエージェント起動

※ 64ビットのPCではエージェントが正常に動作しない可能性があります！

※「libbats.so.0が見つからない」といったエラーが発生した場合は，

- ホームディレクトリの「.bashrc」を開き，下記の文字列を追加
  - 「export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib」
- source ~/.bashrc

linux操作に関しては下記サイトを参照すると良いかも

- http://itpro.nikkeibp.co.jp/article/COLUMN/20070613/274690/

サンプルエージェントについて

☆ 初期状態での各プレイヤの挙動

- キーパー（背番号1）
  - keeper_pc.xmlで定義
  - ゴール前でボール側を向き待機する
- アタッカー（背番号2，3）
  - attacker_pc.xmlで定義
  - 2人でメインとサポートを動的に切り替え
  - メインは相手ゴールに向かってドリブル．サポートはボール後方で待機
  - 他アタッカーとsayでやり取り．自分とボールまでの距離を喋る
- ディフェンダー（背番号4，5，6）
  - defender_pc.xmlで定義
  - 基本的にホームポジションで待機し，ボールが接近してきたら蹴り返す

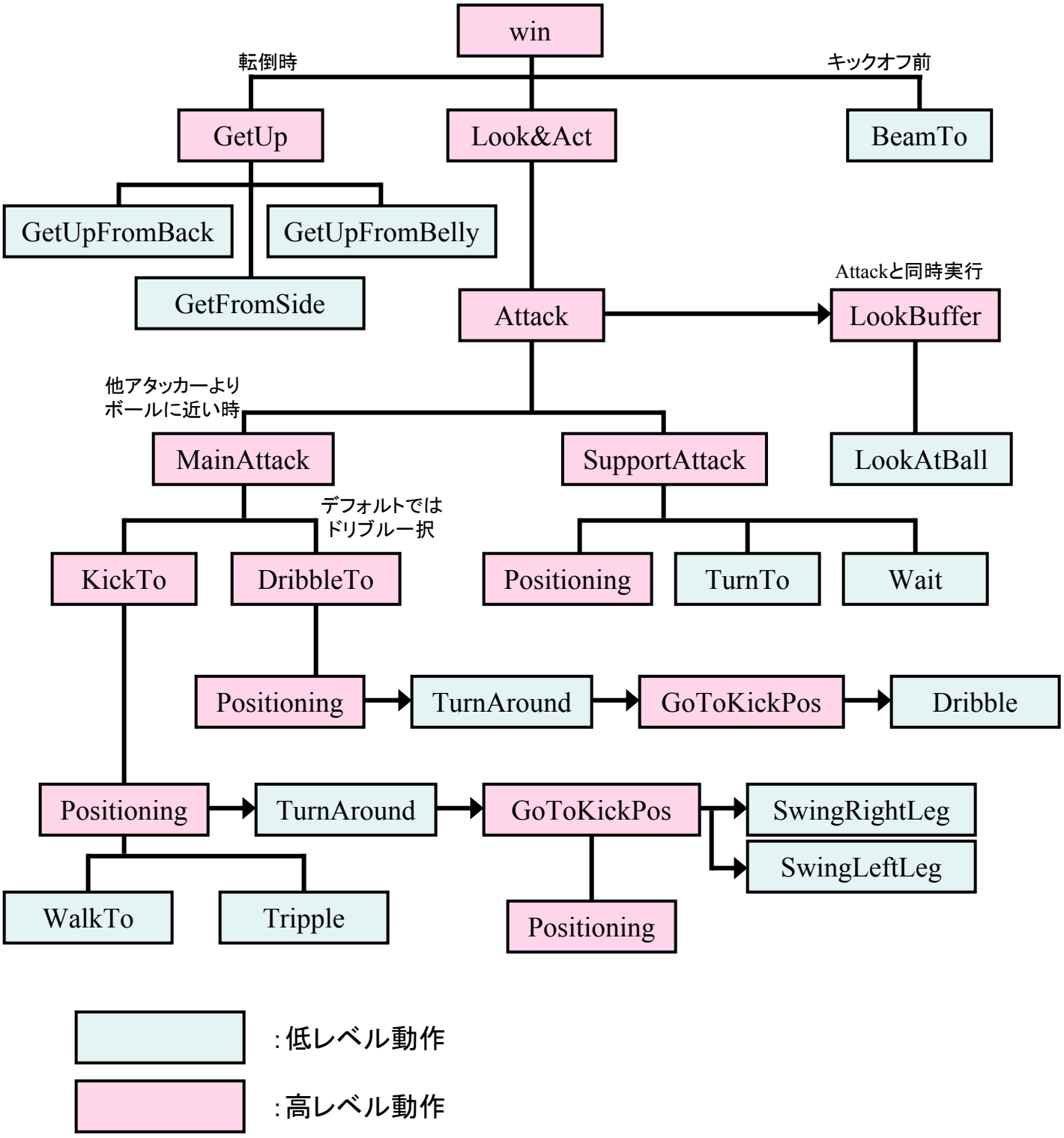
☆ 低レベル動作

- 「歩行」「キック」といった具体的な動作
- confincフォルダ内のxmlファイルでタイプ・パラメータを指定
- 2タイプの動きが可能
  - **Sine**
    - 式「$A\sin(2\pi t/T+B)+C$」に従って関節角度を動作させる
    - xmlファイルでA, B, C, Tの値を指定
    - 主に歩行動作全般に利用
  - **ScriptPlayer**
    - 複数のポーズを指定し，一連の動作として行う
    - xmlファイルで各ポーズの関節角度を指定
    - 起き上がり，キック等の動作に利用

☆ 高レベル動作

- 「ボールに接近」「相手ゴール側を向いてキック」といった行動
- attacker_pc.xml，defender_pc.xml, keeper_pc.xml等で指定
- 各Behaviorのの generateGoal.cc，getCapability.ccにより詳細を決定
  - **generateGoal.cc**
    - 動作の目的（移動先，方向転換の角度など）を与える
    - 設定した目的は，より下位の動作で読み出され使用される
  - **getCapability.cc**
    - 動作の信頼度を与える
    - xmlファイル内で設定することも可能

アタッカーの行動木の例



attacker_pc.xml, confinc/commonbehavior.xmlを参照

チーム開発

☆ 編集するファイル

- 動作の優先順位を変更する場合は**getCapability**
- 動作の目的を変更する場合は**generateGoal**
- getCapability, generateGoalはBehavior内の各フォルダにある
- 動き自体を変更する場合はconfinc内の**各動作.xml**
- 行動木を変更する場合は**attacker_pc.xml**，**commonbehavior.xml**等

☆ ゲーム情報の取得

- **WorldModel**, **AgentModel**, **Localizer**等を使う
  - **WorldModel**
    - 試合時間，スコア，味方サイド等の情報を持つ
  - **AgentModel**
    - 関節角度，背番号，プレイヤタイプ等の情報を持つ
  - **Localizer**
    - 他オブジェクトの位置情報を持つ
    - ローカル座標系，グローバル座標系それぞれでの位置を取得可
- 詳しくは，libbats/docs/html/index.htmlを参照

☆ 編集してみる

- アタッカーがドリブルの代わりにキックで攻めるようにしてみる
  - ドリブル(DribbleTo)の信頼度を下げるorキック(KickTo)の信頼度を上げる
  - 信頼度の値はxmlファイル内で記述
- ゴール周辺ではドリブルではなくシュートを狙うようにしてみる
  - キック(KickTo)の信頼度をゴール周辺でのみ上げる
  - 条件付きで信頼度の値を変更する場合は，該当動作のgetCapabilityを編集
- サポートアタッカーやディフェンダーの待機位置を変えてみる
  - サポートアタッカーの待機位置はSupportAttackのgenerateGoalで設定
  - ディフェンダーの待機位置はdefender_pc.xmlで設定
  - 複雑な動作をさせるにはgenerateGoalで設定する必要がある
- まっすぐゴールを狙うのではなく，サイドから攻めるようにしてみる
  - ドリブル・キックの目標位置を状況に応じて切り替える
- etc

おまけ

- 新しい行動を追加する方法（例えば「Jump」という行動を追加する場合）
  - ./util/createbehavior.pl Jump
    - Behaviorフォルダ内にJump/Jump.cc, Jump/generateGoal.ccなどが生成される
  - OpuciPlayer/init.cc, OpuciPlayer/opuciplayer.hh, bootstrap.shを編集
    - 他のBehaviorと同様の記述を追加
  - ./bootstrap clean; ./bootstrap; ./configure; make でコンパイル
  - 各xmlファイルでbehavior type="Jump"が使用可能に

- デバッガ（VisualDebugger3D）を使う
  - パッケージqt3-dev-toolsが必要
  - VisualDebugger3Dフォルダ内でqmake; makeでコンパイル
  - あらかじめVisualDebugger3Dを起動し，接続待ち状態にしておく
  - エージェントを-dオプションを付けて実行すれば，自動的に接続されデバッガが表示される
  - 基本的に不具合は仕様
  - エージェント側でVD3Connectorを使って情報を埋め込み，参照することができる
  - デバッガに-dオプションを付けて実行すれば，直前のエージェントの動作履歴を確認可

- libbats-2.0-opuci版の，オリジナルlibbats-2.0からの変更点
  - Behaviorにcreate関数を追加
  - Types::nameOf関数を拡張
  - Cochlea::intergrate関数内のhear解釈部分を有効に
  - ほかにもあるかも

- libbats-2.0を使う上での注意点
  - leg2，leg3がサーバマニュアルと比較して逆になっている(leg2が前後，leg3が左右に動作)
  - 角度は全てラジアンで処理

- Eigen2について
  - 配列，ベクトル，行列などを扱うライブラリ
  - 参考：http://eigen.tuxfamily.org/dox-devel/QuickRefPage.html