Pbalite Code Organization

(intended to be unchanged by students)

Data Representation

Vector

3D vectors with overloaded operators for linear algebra
- Dot product
- Cross product
- Add, subtract
- Multiply, divide (by scalar)

Color

4 component color object with overloaded operators
- Add
- Multiply, divide (by scalar and by color)

Matrix

3X3 matrix with overloaded operators for linear algebra
- Vector times matrix, matrix times vector
- Matrix time matrix
- Inverse, transpose
- Add, subtract
- Exponential, determinant

Helpers

LinearAlgebra

PbaUtils

Viewing/Running Animation

PbaViewer

Creates display window
Runs main loop (infinite loop)
Refreshes display and idle actions
Processes events and directs them to PbaThings
Uses GLUT for window/event managements
Runs OpenGL inside window

Animation Framework

PbaThing

Base class for animatable scenes
Processes events from viewer
Passes events to derived objects
Triggers animation update during idle period
Triggers display update during display period

ScreenCapturePPM

Derives from PbaThing
Captures screen image during display update
Writes captured image to an ascii PPM image file
Does not alter screen content

```
The command
   > make
compiles these and other code into the library
   lib/libpba.a
```

Student Adjustable

(in directory 'things')

MyThing

Derives from PbaThing
Holds animatable data
Updates animation during idle period
Updates display during display period
Processes keyboard events
Provides information
Central focus of your effort

pbalitesim

Main file for executable
Invokes PbaViewer
Invokes MyThing
Invokes ScreenCapturePPM
Lets viewer ingest MyThing & ScreenCapturePPM
Starts main loop
You may never need to change this code

???

Additional files you may wish to create
To compile any of these files, you will need to modify the Makefile in the top directory

Student Adjustable

(in directory 'things')

MyThing

Derives from PbaThing
Holds animatable data
Updates animation during idle period
Updates display during display period
Processes keyboard events
Provides information
Central focus of your effort

pbalitesim

Main file for executable
Invokes PbaViewer
Invokes MyThing
Invokes ScreenCapturePPM
Lets viewer ingest MyThing & ScreenCapturePPM
Starts main loop
You may never need to change this code

???

Additional files you may wish to create
To compile any of these files, you will need to modify the Makefile in the top directory

MyThing Methods & Data

```
void Init( const std::vector<std::string>& args );


//! Implements a display event
//! This is where you code the opengl calls to display
//! your system.
void Display();

//! Implements responses to keyboard events
//! This is called when you hit a key
void Keyboard( unsigned char key, int x, int y );

//! Implements simulator updates during an idle period
//! This is where the update process is coded
//! for your dynamics problem.
void solve();

//! Implements reseting parameters and/or state
//! This is called when you hit the 'r' key
void Reset();

//! Displays usage information on stdout
//! If you set up actions with the Keyboard()
//! callback, you should include a statement
//! here as to what the keyboard option is.
void Usage();
```

```
class ParticleState
{
  public:
    ParticleState();
   ~ParticleState(){};

   Vector position;
   Vector velocity;
   Color color;
   float mass;
};

// This is all of the particles in the system
std::vector<ParticleState> particles;
```

Student Adjustable

(in directory 'things')

MyThing

Derives from PbaThing
Holds animatable data
Updates animation during idle period
Updates display during display period
Processes keyboard events
Provides information
Central focus of your effort

pbalitesim

Main file for executable
Invokes PbaViewer
Invokes MyThing
Invokes ScreenCapturePPM
Lets viewer ingest MyThing & ScreenCapturePPM
Starts main loop
You may never need to change this code

???

Additional files you may wish to create
To compile any of these files, you will need to modify the Makefile in the top directory

MyThing Methods & Data

```
void Init( const std::vector<std::string>& args );


//! Implements a display event
//! This is where you code the opengl calls to display
//! your system.
void Display();

//! Implements responses to keyboard events
//! This is called when you hit a key
void Keyboard( unsigned char key, int x, int y );

//! Implements simulator updates during an idle period
//! This is where the update process is coded
//! for your dynamics problem.
void solve();

//! Implements reseting parameters and/or state
//! This is called when you hit the 'r' key
void Reset();

//! Displays usage information on stdout
//! If you set up actions with the Keyboard()
//! callback, you should include a statement
//! here as to what the keyboard option is.
void Usage();
```

```
class ParticleState
{
  public:
    ParticleState();
   ~ParticleState(){};

   Vector position;
   Vector velocity;
   Color color;
   float mass;
};

// This is all of the particles in the system
std::vector<ParticleState> particles;
```

```
The command
    > make sim
compiles things/pbalitesim.C, links with lib/libpba.a, and creates executable
    pbalitesim
```

Makefile Build Options

> make clean	Delete: .o files lib/libpba.a pbalitesim
> make	Compile .o files listed in Makefile, including everything in base/ Assembles library lib/libpba.a
>make sim	Compile things/pbalitesim.C and link lib/libpba.a Generate executable pbalitesim

Pbalite Directories

- **include**: Header fi les for provided code
- **base**: C++ implemented of provided code
- **things**: location for your implementation; example implementation
- **lib**: location of libpba.a
- **models**: assorted obj fi les that might prove handy
- **python**: some scripts that are potentially handy