Appendix D

Source Code for Increment 1

ARCGMOBJ.H

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                     /
// Copyright (C) 1996 Software Architects,                 /
// All rights reserved.                                     /
//                                                          /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                              /
/////////////////////////////////////////////////////////////
#ifndef __ARCGMOBJ_H__
#define __ARCGMOBJ_H__

/*
   This is the root of the hierarchy of objects that
   interact in the game.  All objects have bitmaps
   that are drawn to identify the object and a bounding
   rectangle that is used to identify collisions with
   other objects.
*/
////////////////////////////////////////////////////////////////////////////////
#include "stdafx.h"

class Bitmap;
class CRect;

class ArcadeGameObject  : public CObject
{
public:
   ArcadeGameObject();
   ArcadeGameObject(CBitmap*);
   virtual ~ArcadeGameObject();

public:
   virtual CRect getBoundingRect()const;
   virtual void   setBoundingRect(const CRect& newRect);
   virtual CBitmap* getBitmap() ;
   //virtual void setBitmap(CBitmap* bitmap);
   virtual CBitmap* getBitmap() const;
   virtual void tick()=0;



//Attributes
protected:
   CRect_boundingRect;// bounding rectangle of the game object
   CBitmap*_objBitmap;// bitmap representing the game object'

};

////////////////////////////////////////////////////////////////////////////////

#endif // __ARCGMOBJ_H__
```

ARCGMOBJ.CPP

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                     /
// Copyright (C) 1996 Software Architects,                 /
// All rights reserved.                                     /
//                                                          /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                              /
/////////////////////////////////////////////////////////////
// arcgmobj.cpp: Defines the class behaviors for the abstract
//         class ArcadeGameObject.
//
#include "stdafx.h"
#include "arcgmobj.h"



////////////////////////////////////////////////////////////////////////////////
// ArcadeGamePiece creation
```

```
ArcadeGameObject::ArcadeGameObject()
{
  _objBitmap = new CBitmap;
}

ArcadeGameObject::ArcadeGameObject(CBitmap* bitMapPtr)
{
   _objBitmap = bitMapPtr;
}

ArcadeGameObject::~ArcadeGameObject()
{
  delete _objBitmap;
}

CRect ArcadeGameObject::getBoundingRect() const
{
  return _boundingRect;
}

void ArcadeGameObject::setBoundingRect(const CRect& newRect)
{
  _boundingRect = newRect;
}

CBitmap* ArcadeGameObject::getBitmap()
{
  return _objBitmap;
}

//void ArcadeGameObject::setBitmap (CBitmap* newBitmap)
//{
// _objBitmap = newBitmap;
//}

CBitmap* ArcadeGameObject::getBitmap() const{
  return _objBitmap;
}
```

ARCGMPC.H

```
////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                    /
// Copyright (C) 1996 Software Architects,                /
// All rights reserved.                                   /
//                                                        /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                            /
////////////////////////////////////////////////////////////
#ifndef __ARCGMPC_H__
#define __ARCGMPC_H__

/*
  An arcade game piece is a component of an arcade game that can
  appear on a playing field.  As such, every arcade game piece has
  a position within the field and a "footprint" within the field.
  Associated with the footprint is a bounding rectangle.
*/
////////////////////////////////////////////////////////////////////////////
#include "stdafx.h"
#include "arcgmobj.h"

class CPoint;
class PlayingField;
//class Puck;
//class Trajectory;

class ArcadeGamePiece:public ArcadeGameObject
{
public:
  ArcadeGamePiece();
  ArcadeGamePiece(const CPoint&);
  ArcadeGamePiece(PlayingField*, const CPoint&);
  ArcadeGamePiece(ArcadeGamePiece&);
  ~ArcadeGamePiece();
```

```
public:


   CPoint getPosition();
   void   setPosition(CPoint);
   PlayingField* getPlayingField();

//Attributes
protected:
   CPoint_posField;// position in field
   PlayingField* _fieldPtr;

};

/////////////////////////////////////////////////////////////////////////////

#endif // __ARCGMPC_H__
```

ARCGMPC.CPP

```
/////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                /
// Copyright (C) 1996 Software Architects,            /
// All rights reserved.                               /
//                                                    /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                        /
/////////////////////////////////////////////////////////
// arcgmpc.cpp: Defines the class behaviors for the abstract
//         class ArcadeGamePiece.
//
#include "stdafx.h"
#include "arcgmpc.h"


/////////////////////////////////////////////////////////////////////////////
```

```
// ArcadeGamePiece creation


ArcadeGamePiece::ArcadeGamePiece()
{
}

ArcadeGamePiece::ArcadeGamePiece (const CPoint& position)
{
   _posField = position;
}


ArcadeGamePiece::ArcadeGamePiece(PlayingField* fieldPtr, const CPoint&
            referencePoint):
   _posField(referencePoint){
   _fieldPtr = fieldPtr;
}

ArcadeGamePiece::ArcadeGamePiece(ArcadeGamePiece& aPiece){
   _posField = aPiece._posField;
   _fieldPtr = aPiece._fieldPtr;
}

ArcadeGamePiece::~ArcadeGamePiece(){}

CPoint ArcadeGamePiece::getPosition(){
   return _posField;
}

void ArcadeGamePiece::setPosition(CPoint position){
   _posField = position;
}


PlayingField* ArcadeGamePiece::getPlayingField(){
   return _fieldPtr;
}
```

BRICKDOC.H

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                     /
// Copyright (C) 1996 Software Architects,                 /
// All rights reserved.                                    /
//                                                         /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                             /
/////////////////////////////////////////////////////////////
// brickdoc.h : interface of the CBricklesDoc class        /
//                                                         /
/////////////////////////////////////////////////////////////

class Puck;
class Paddle;
class PlayingField;
class Timer;
class Mouse;

class CBricklesDoc : public CDocument
{
protected: // create from serialization only
   CBricklesDoc();
   DECLARE_DYNCREATE(CBricklesDoc)


// Attributes
public:
// Operations
public:
   CView* getView();
   void displayStatusMessage(LPCSTR);
   void setBoundingRect(CRect);
//   CRect getBoundingRect();
   Puck* getPuck();
   Paddle* getPaddle();
   Mouse* getMouse();
   void tick();

// Implementation
public:
   virtual ~CBricklesDoc();
   virtual void Serialize(CArchive& ar);   // overridden for document i/o
#ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump(CDumpContext& dc) const;
#endif

protected:
   virtual BOOL OnNewDocument();
   //CRect _boundingBox;
   Puck* _currentPuck;
   Paddle* _thePaddle;
   PlayingField* _fieldPtr;
   Timer* _theTimer;
   Mouse* _theMouse;

// Generated message map functions
protected:
   //{{AFX_MSG(CBricklesDoc)
      // NOTE - the ClassWizard will add and remove member functions here.
      //    DO NOT EDIT what you see in these blocks of generated code !
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

BRICKDOC.CPP

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                     /
// Copyright (C) 1996 Software Architects,                 /
// All rights reserved.                                    /
//                                                         /
```

```
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                              /
/////////////////////////////////////////////////////////////
// brickdoc.cpp : implementation of the CBricklesDoc class
//

#include “stdafx.h”
#include “brickles.h”
#include “puck.h”
#include “paddle.h”
#include “plyngfld.h”
#include “brickvw.h”
#include “timer.h”
#include “mouse.h”

#include “brickdoc.h”

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CBricklesDoc

IMPLEMENT_DYNCREATE(CBricklesDoc, CDocument)

BEGIN_MESSAGE_MAP(CBricklesDoc, CDocument)
   //{{AFX_MSG_MAP(CBricklesDoc)
      // NOTE - the ClassWizard will add and remove mapping macros here.
      //    DO NOT EDIT what you see in these blocks of generated code!
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CBricklesDoc construction/destruction

CBricklesDoc::CBricklesDoc()
{
   // TODO: add one-time construction code here

}

CBricklesDoc::~CBricklesDoc()
{
}

void CBricklesDoc::setBoundingRect(CRect boundingBox){
   _fieldPtr->setBoundingRect(boundingBox);
//  _currentPuck->setBoundingRect(boundingBox);
//  _thePaddle->setBoundingRect(boundingBox);

}


Puck* CBricklesDoc::getPuck(){
   return _currentPuck;
}

Paddle* CBricklesDoc::getPaddle(){
   return _thePaddle;
}

Mouse* CBricklesDoc::getMouse(){
   return _theMouse;
}

void CBricklesDoc::tick(){
   //_theTimer->tick();
   _fieldPtr->tick();
   _currentPuck->tick();
}

CView*  CBricklesDoc::getView(){
   POSITION pos = GetFirstViewPosition();
   return this->GetNextView(pos);
}

/*  This method displays a message in the Status Bar at the
   bottom of the game’s window.  It can be used anytime after
```

```
   the view is created.
*/
void CBricklesDoc::displayStatusMessage(LPCSTR message){
   ((CBricklesView*)this->getView())->displayMessage(message);
}


BOOL CBricklesDoc::OnNewDocument()
{
   if (!CDocument::OnNewDocument())
      return FALSE;

   // TODO: add reinitialization code here
   // (SDI documents will reuse this document)
   _fieldPtr = new PlayingField();//this);
   _currentPuck = new Puck(_fieldPtr);
   _thePaddle = new Paddle(_fieldPtr);
   _theMouse = new Mouse();
   //_theMouse->hideCursor();
   _theMouse->enroll(_thePaddle);
   _theMouse->enroll(_thePaddle);
   //_theTimer = new Timer();
   //_theTimer->enroll(_fieldPtr);
   //_theTimer->enroll(_currentPuck);
   //_theTimer->enroll(_thePaddle);
   return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CBricklesDoc serialization

void CBricklesDoc::Serialize(CArchive& ar)
{
   if (ar.IsStoring())
   {
      // TODO: add storing code here
   }
   else
   {
      // TODO: add loading code here
   }
}

/////////////////////////////////////////////////////////////////////////////
// CBricklesDoc diagnostics

#ifdef _DEBUG
void CBricklesDoc::AssertValid() const
{
   CDocument::AssertValid();
}

void CBricklesDoc::Dump(CDumpContext& dc) const
{
   CDocument::Dump(dc);
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CBricklesDoc commands
```

BRICKLES.H

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                      /
// Copyright (C) 1996 Software Architects,                  /
// All rights reserved.                                     /
//                                                          /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                              /
/////////////////////////////////////////////////////////////
// brickles.h : main header file for the BRICKLES application
//

#ifndef __AFXWIN_H__
   #error include 'stdafx.h' before including this file for PCH
#endif
```

```
#include "resource.h"       // main symbols

/////////////////////////////////////////////////////////////////////////////
// CBricklesApp:
// See brickles.cpp for the implementation of this class
//

class CBricklesApp : public CWinApp
{
public:
   CBricklesApp();

// Overrides
   virtual BOOL InitInstance();

// Implementation

   //{{AFX_MSG(CBricklesApp)
   afx_msg void OnAppAbout();
      // NOTE - the ClassWizard will add and remove member functions here.
      //    DO NOT EDIT what you see in these blocks of generated code !
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};


/////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About
class CAboutDlg : public CDialog
{
public:
   CAboutDlg();

// Dialog Data
   //{{AFX_DATA(CAboutDlg)
   enum { IDD = IDD_PLAYBOX };
   //}}AFX_DATA

// Implementation
protected:
   virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
   //{{AFX_MSG(CAboutDlg)
      // No message handlers
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};
```

BRICKLES.CPP

```
/////////////////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                /
// Copyright (C) 1996 Software Architects,            /
// All rights reserved.                               /
//                                                    /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                        /
/////////////////////////////////////////////////////////////////////////
// brickles.cpp : Defines the class behaviors for the application.
//

#include "stdafx.h"
#include "brickles.h"

#include "mainfrm.h"
#include "brickdoc.h"
#include "brickvw.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CBricklesApp
```

```
BEGIN_MESSAGE_MAP(CBricklesApp, CWinApp)
   //{{AFX_MSG_MAP(CBricklesApp)
   ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
      // NOTE - the ClassWizard will add and remove mapping macros here.
      //    DO NOT EDIT what you see in these blocks of generated code!
   //}}AFX_MSG_MAP
   // Standard file based document commands
   ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
   ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CBricklesApp construction

CBricklesApp::CBricklesApp()
{
   // TODO: add construction code here,
   // Place all significant initialization in InitInstance
}

/////////////////////////////////////////////////////////////////////////////
// The one and only CBricklesApp object

CBricklesApp NEAR theApp;

/////////////////////////////////////////////////////////////////////////////
// CBricklesApp initialization

BOOL CBricklesApp::InitInstance()
{
   // Standard initialization
   // If you are not using these features and wish to reduce the size
   //  of your final executable, you should remove from the following
   //  the specific initialization routines you do not need.

   SetDialogBkColor();        // Set dialog background color to gray
   LoadStdProfileSettings();  // Load standard INI file options (including MRU)

   // Register the application's document templates.  Document templates
   //  serve as the connection between documents, frame windows and views.

   CSingleDocTemplate* pDocTemplate;
   pDocTemplate = new CSingleDocTemplate(
      IDR_MAINFRAME,
      RUNTIME_CLASS(CBricklesDoc),
      RUNTIME_CLASS(CMainFrame),     // main SDI frame window
      RUNTIME_CLASS(CBricklesView));
   AddDocTemplate(pDocTemplate);

   // create a new (empty) document
   OnFileNew();

   if (m_lpCmdLine[0] != '\0')
   {
      // TODO: add command line processing here
   }


   return TRUE;
}

// App command to run the dialog
void CBricklesApp::OnAppAbout()
{
//  CAboutDlg aboutDlg;
//  aboutDlg.DoModal();
}


CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
   //{{AFX_DATA_INIT(CAboutDlg)
   //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
   CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CAboutDlg)
   //}}AFX_DATA_MAP
```

```
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
   //{{AFX_MSG_MAP(CAboutDlg)
      // No message handlers
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()


/////////////////////////////////////////////////////////////////////////////
// CBricklesApp commands
```

BRICKVW.H

```
//////////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                    /
// Copyright (C) 1996 Software Architects,                 /
// All rights reserved.                                    /
//                                                         /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                             /
//////////////////////////////////////////////////////////////////
// brickvw.h : interface of the CBricklesView class
//
/////////////////////////////////////////////////////////////////////////////

#include "stdafx.h"

class CBricklesView : public CView
{
protected: // create from serialization only
   CBricklesView();
   DECLARE_DYNCREATE(CBricklesView)

// Attributes
public:
   CBricklesDoc* GetDocument();
   virtual void displayMessage(LPCSTR message);
   static void display(LPCSTR message);
// Operations
public:

// Implementation
public:
   virtual ~CBricklesView();
   virtual void OnDraw(CDC* pDC);  // overridden to draw this view
   void OnInitialUpdate();
#ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump(CDumpContext& dc) const;
#endif

protected:
// int boxInit;

// Generated message map functions
protected:
   //{{AFX_MSG(CBricklesView)
   afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
   afx_msg void OnTimer(UINT nIDEvent);
   afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
   afx_msg void OnMouseMove(UINT nFlags, CPoint point);
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()

private:
   static CBricklesView* theBricklesViewPtr;
   int _timerID;
};

#ifndef _DEBUG  // debug version in brickvw.cpp
inline CBricklesDoc* CBricklesView::GetDocument()
  { return (CBricklesDoc*)m_pDocument; }
#endif

/////////////////////////////////////////////////////////////////////////////
```

BRICKVW.CPP

```
///////////////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                        /
// Copyright (C) 1996 Software Architects,                    /
// All rights reserved.                                       /
//                                                            /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                                /
///////////////////////////////////////////////////////////////////////
// brickvw.cpp : implementation of the CBricklesView class
//

#include "stdafx.h"
#include "brickles.h"
#include "puck.h"
#include "paddle.h"
#include "mainfrm.h"
#include "brickdoc.h"
#include "brickvw.h"
#include "resource.h"
#include "mouse.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CBricklesView

IMPLEMENT_DYNCREATE(CBricklesView, CView)

BEGIN_MESSAGE_MAP(CBricklesView, CView)
   //{{AFX_MSG_MAP(CBricklesView)
   ON_WM_CREATE()
   ON_WM_TIMER()
   ON_WM_LBUTTONDOWN()
   ON_WM_MOUSEMOVE()
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CBricklesView construction/destruction



CBricklesView* CBricklesView::theBricklesViewPtr = 0;

CBricklesView::CBricklesView()
{
   // TODO: add construction code here
   CBricklesView::theBricklesViewPtr = this;
}

CBricklesView::~CBricklesView()
{
   // CBricklesDoc* pDoc = GetDocument();
      int tmp = this->KillTimer(_timerID);
   // (pDoc->getMouse())->showCursor();
}

/////////////////////////////////////////////////////////////////////////////
// CBricklesView drawing

void CBricklesView::OnDraw(CDC* pDC)
{
   CBricklesDoc* pDoc = GetDocument();
   ASSERT_VALID(pDoc);

   CBitmap* pOldBitmap;
   CDC dcMem;
   dcMem.CreateCompatibleDC(pDC);

   Puck* currentPuck;
   Paddle* thePaddle;
   currentPuck = pDoc->getPuck();
   thePaddle = pDoc->getPaddle();
   CPoint puckPosition = currentPuck->getPosition();
```

```
    CPoint paddlePosition = thePaddle->getPosition();
    pOldBitmap = dcMem.SelectObject(currentPuck->getBitmap());
    pDC->BitBlt(puckPosition.x,puckPosition.y,20,20,&dcMem,0,0,SRCCOPY);
    dcMem.SelectObject(pOldBitmap);
    pOldBitmap = dcMem.SelectObject(thePaddle->getBitmap());
    pDC->BitBlt(paddlePosition.x,paddlePosition.y,60,20,&dcMem,0,0,SRC-
            COPY);
    dcMem.SelectObject(pOldBitmap);
   dcMem.DeleteDC();
   ReleaseDC(pDC);
}

/////////////////////////////////////////////////////////////////////////////
// CBricklesView diagnostics

#ifdef _DEBUG
void CBricklesView::AssertValid() const
{
   CView::AssertValid();
}

void CBricklesView::Dump(CDumpContext& dc) const
{
   CView::Dump(dc);
}

CBricklesDoc* CBricklesView::GetDocument() // non-debug version is inline
{
   ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CBricklesDoc)));
   return (CBricklesDoc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CBricklesView message handlers

int CBricklesView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
   if (CView::OnCreate(lpCreateStruct) == -1)
      return -1;

   _timerID = this->SetTimer(1, 20 , NULL);

   if (!_timerID)
   {
      AfxMessageBox("Not enough timers available for this window.");
            // signal creation failure...
      return -1;
   }

   return 0;
}


void CBricklesView::OnTimer(UINT nIDEvent)
{
   CBricklesDoc* pDoc = GetDocument();

   pDoc->tick();
   Invalidate();
   CView::OnTimer(nIDEvent);
}

void CBricklesView::OnLButtonDown(UINT nFlags, CPoint point)
{
   int Return ;
   CAboutDlg PlayDlg;
   CBricklesDoc* pDoc = GetDocument();

   int tmp = this->KillTimer(_timerID);
   //(pDoc->getMouse())->showCursor();
   Return = PlayDlg.DoModal();
   if (Return == IDOK)
   {
//     (pDoc->getMouse())->hideCursor();
      _timerID = this->SetTimer(1, 20 , NULL);
      if (!_timerID)
      {
         this->MessageBox("Not enough timers available for this window.",
               "Brickles", MB_ICONEXCLAMATION | MB_OK);
```

```
      }
   }
   else {
      this->Detach();
   }

   //CView::OnLButtonDown(nFlags, point);
}

void CBricklesView::OnInitialUpdate(){
   CBricklesDoc* pDoc = GetDocument();
   CRect boundingBox;
   GetClientRect(boundingBox);
   pDoc->setBoundingRect(boundingBox);
}


void CBricklesView::display(LPCSTR message) {

   CBricklesView::theBricklesViewPtr->displayMessage(message);

}

void CBricklesView::displayMessage(LPCSTR message)
{
   CMainFrame*mainframe;
   mainframe = (CMainFrame*)this->GetParentFrame();
   mainframe->setStatusMessage(message);
}


void CBricklesView::OnMouseMove(UINT nFlags, CPoint point)
{
   // TODO: Add your message handler code here and/or call default


   CBricklesDoc* pDoc = GetDocument();
   (pDoc->getMouse())->setPosition(point);
   CView::OnMouseMove(nFlags, point);
}
```

MAINFRM.H

```
//////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                  /
// Copyright (C) 1996 Software Architects,              /
// All rights reserved.                                 /
//                                                      /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                          /
//////////////////////////////////////////////////////////
// mainfrm.h : interface of the CMainFrame class
//
/////////////////////////////////////////////////////////////////////////////

class CMainFrame : public CFrameWnd
{
protected: // create from serialization only
   CMainFrame();
   DECLARE_DYNCREATE(CMainFrame)
   CStatusBar  m_wndStatusBar;
   CToolBar    m_wndToolBar;

// Attributes
public:
   void setStatusMessage(LPCSTR aString);
// Operations
public:

// Implementation
public:
   virtual ~CMainFrame();
#ifdef _DEBUG
   virtual void AssertValid() const;
   virtual void Dump(CDumpContext& dc) const;
#endif
```

```
// Generated message map functions
protected:
   //{{AFX_MSG(CMainFrame)
   afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
   //}}AFX_MSG
   DECLARE_MESSAGE_MAP()
};

/////////////////////////////////////////////////////////////////////////////
```

MAINFRM.CPP

```
///////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                   /
// Copyright (C) 1996 Software Architects,               /
// All rights reserved.                                  /
//                                                       /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                           /
///////////////////////////////////////////////////////////
// mainfrm.cpp : implementation of the CMainFrame class
//

#include "stdafx.h"
#include "brickles.h"

#include "mainfrm.h"

#ifdef _DEBUG
#undef THIS_FILE
static char BASED_CODE THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
   //{{AFX_MSG_MAP(CMainFrame)
   ON_WM_CREATE()
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

static UINT BASED_CODE indicators[] =
{
   ID_SEPARATOR,           // status line indicator
   ID_INDICATOR_CAPS,
   ID_INDICATOR_NUM,
   ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
   // TODO: add member initialization code here
}

CMainFrame::~CMainFrame()
{
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
   CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
   CFrameWnd::Dump(dc);
```

```
}

#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
   if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
      return -1;

   // TODO: Add your specialized creation code here
   lpCreateStruct->style = WS_CAPTION | WS_BORDER | WS_VISIBLE
                              | WS_MAXIMIZE;
   lpCreateStruct->hMenu = NULL;
   if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
      return -1;

   if (!m_wndStatusBar.Create(this) ||
      !m_wndStatusBar.SetIndicators(indicators,
        sizeof(indicators)/sizeof(UINT)))
   {
      TRACE("Failed to create status bar\n");
      return -1;      // fail to create
   }
   m_hMenuDefault = NULL;
   ShowWindow(SW_SHOWMAXIMIZED);

   return 0;
}

void CMainFrame::setStatusMessage(LPCSTR aString)
{
   m_wndStatusBar.SetPaneText(0,(const char far*) aString);
}
```

MOUSE.H

```
///////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                   /
// Copyright (C) 1996 Software Architects,               /
// All rights reserved.                                  /
//                                                       /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                           /
///////////////////////////////////////////////////////////
//mouse.h
//

class CPoint;
class MovablePiece;

class Mouse{
   public:
      Mouse();
      ~Mouse();

      CPoint& getPosition();
      void setPosition(CPoint&);
      void enroll(MovablePiece*)const;
      void withdraw(MovablePiece*)const;
      void enable();
      void disable();
      void hideCursor();
      void showCursor();

      BOOL isEnabled() const;
      BOOL cursorIsShowing() const;

   protected:
      BOOL _enabled;
      BOOL _showing;
      CPoint  _currentPosition;
      MovablePiece* _enrollees[5];
      int _enrolleeCount;
```

```
        int max;


};
```

MOUSE.CPP

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                      /
// Copyright (C) 1996 Software Architects,                  /
// All rights reserved.                                     /
//                                                          /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                              /
/////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "mouse.h"
#include "mvblpc.h"

    Mouse::Mouse(){
        _enrolleeCount = 0;
        max = 5;
    }

    Mouse::~Mouse(){}

    CPoint& Mouse::getPosition(){
        return _currentPosition;
    }

    void Mouse::setPosition(CPoint& currentPosition){
        _currentPosition = currentPosition;
        for(int i=0;i<_enrolleeCount; i++){
            _enrollees[i]->move(currentPosition);
        }
    }

void Mouse::enroll(MovablePiece* item)const{
    if (_enrolleeCount <= max){
        _enrollees[_enrolleeCount]= item;
        ((Mouse*)this)->_enrolleeCount = ((Mouse*)this)->_enrolleeCount +
        1;
    }

}


void Mouse::withdraw(MovablePiece* item)const{
    int i = 0;
    while (i < _enrolleeCount && _enrolleeCount > 0){
        if (item = _enrollees[i]){
            if (i < _enrolleeCount-1){
                _enrollees[i] = _enrollees[_enrolleeCount];
            }
            ((Mouse*)this)->_enrolleeCount = ((Mouse*)this)->_enrolleeCount
        - 1;
        }
    }
}

void Mouse::enable(){
    EnableHardwareInput(TRUE);
    _enabled=TRUE;
}

void Mouse::disable(){
    EnableHardwareInput(FALSE);
    _enabled=FALSE;
}

void Mouse::hideCursor(){
    ShowCursor(FALSE);
    _showing = FALSE;
}

void Mouse::showCursor(){
    ShowCursor(TRUE);
```

```
    _showing = TRUE;
  }

  BOOL Mouse::isEnabled() const{
    return _enabled;
  }


  BOOL Mouse::cursorIsShowing() const{
    return _showing;
  }
```

MVBLPC.H

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                      /
// Copyright (C) 1996 Software Architects,                  /
// All rights reserved.                                     /
//                                                          /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                              /
/////////////////////////////////////////////////////////////

#ifndef MVBLPC_H

#define MVBLPC_H

/*

   An movable piece is an arcade game piece that can move in

   a playing field during a match.  Each movable piece has a

   velocity which is used to determine its trajectory over

   a unit interval of time.

   The position of a movable object in a playing field is the

   point that "sticks out" the furthest in the direction it

   is traveling.


   A movable piece is not necessarily always in motion.  If

   it is in motion, then it moves when told to do so.  Otherwise,

   move requests are ignored.

*/


#include "stdafx.h"

#include "ArcGmPc.h"

#include "Velocity.h"


//class Puck;

class PlayingField;
class CPoint;


class MovablePiece : public ArcadeGamePiece {

   public:
```

```
    MovablePiece(MovablePiece&);
    MovablePiece();
    ~MovablePiece();

    virtual void initialize();
    virtual Velocity& getVelocity();
    virtual void setVelocity(Velocity&);
    virtual CPoint getLeadingPoint();
    virtual void move()=0;
     virtual void move(CPoint&)=0;
    BOOL isInMotion() const;
    void tick();
    virtual Speed getSpeed();
    virtual Direction getDirection();
    virtual void changeSpeed(int newSpeed);
    virtual void changeDirection(int newDirection);
    virtual void reverseX();
    virtual void reverseY();

  protected:
    MovablePiece(PlayingField*, const CPoint&, const Speed, const Direction);
    Velocity _currentVelocity;
    BOOL _moving;
    CPoint _leadingPoint;
  private:

};

/*
Notes:
  1. Finally, an overloaded member function (resume)!

  2. "move()" computes the trajectory for the next unit interval of
     time (in response to "tick()") and notifies the playing field
     that it has moved.  If the move is accepted (normal return),
     then the position is updated to the endpoint of the trajectory.
     Otherwise, (a Collision exception), the collision is set
     to occur, resulting in the position being set that way.

  3. "Position" might have to be the location of the upper left
```

corner of the bounding rectangle, and the collision point is independent of that. That needs to be settled during implementation!

4. Here's the method I have in mind:

```
void MovablePiece::tick(){
   this->move();
   try {
      (this->playingField).moved(this);
   }
   catch (Collision collision) {
      collision.occur();
   };
}
```

It is a reasonable implementation for at least some specializations.

[HELP: I am not sure how exception objects get deleted...]

5. JM: Will this illustrate the problem you encountered in PACT with handling exceptions in inherited subclass methods? [I don't remember the exact details.]

```
*/

#endif
```

MVBLPC.CPP

```
//////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                       /
// Copyright (C) 1996 Software Architects,                   /
// All rights reserved.                                      /
//                                                           /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                               /
//////////////////////////////////////////////////////////////

#include "stdafx.h"

#include "mvblpc.h"

#include "plyngfld.h"

#include "velocity.h"

     MovablePiece::MovablePiece()
     {
```

```
   initialize();
}

MovablePiece::MovablePiece(PlayingField* fieldPtr, const CPoint& initial-
      Location,

   const Speed newSpeed,const Direction newDirection):

   ArcadeGamePiece(fieldPtr,initialLocation),

   _currentVelocity(newSpeed,newDirection)

{
   this->initialize();
}

MovablePiece::MovablePiece(MovablePiece& oldPiece)

   :ArcadeGamePiece(oldPiece._fieldPtr,oldPiece._posField)

{

   this->initialize();

}

void MovablePiece::initialize(){

   _moving = TRUE;

}

MovablePiece::~MovablePiece(){}

void MovablePiece::tick(){

   this->move();
}

CPoint MovablePiece::getLeadingPoint(){

   Direction _direction = _currentVelocity.getDirection();

   if (_direction >= 0 && _direction < 90)

      _leadingPoint =  _boundingRect.BottomRight();

   else if(_direction >= 90 && _direction <180)

      _leadingPoint =
      CPoint(_boundingRect.TopLeft().x,_boundingRect.BottomRight().y);

   else if(_direction >= 180 && _direction <270)

      _leadingPoint =  _boundingRect.TopLeft();

   else if(_direction >= 270 && _direction <360)

      _leadingPoint = CPoint(_boundingRect.Bottom-
      Right().x,_boundingRect.TopLeft().y);

   return _leadingPoint;

}
```

```
Velocity& MovablePiece::getVelocity() {
   return _currentVelocity;
}

void MovablePiece::setVelocity(Velocity& newVelocity){
   _currentVelocity = newVelocity;
}

void MovablePiece::reverseX(){
   _currentVelocity.reverseX();
}

void MovablePiece::reverseY(){
   _currentVelocity.reverseY();
}

BOOL MovablePiece::isInMotion() const{
   return _moving;
}

Speed MovablePiece::getSpeed() {
   return _currentVelocity.getSpeed();
}

Direction MovablePiece::getDirection() {
   return _currentVelocity.getDirection();
}

void MovablePiece::changeSpeed (int newSpeed){
   _currentVelocity.setSpeed(newSpeed);
}

void MovablePiece::changeDirection (int newDirection){
   _currentVelocity.setDirection(newDirection);
}
```

PADDLE.H

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                    /
// Copyright (C) 1996 Software Architects,                /
```

```
// All rights reserved.                          /
//                                               /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                   /
//////////////////////////////////////////////////////////
// PADDLE.H - Class definition for the paddle class
#ifndef __PADDLE_H__
#define __PADDLE_H__

/////////////////////////////////////////////////////////////////////////////

class PlayingField;

#include "mvblpc.h"

class Paddle : public MovablePiece
{
public:
   Paddle();
   Paddle(PlayingField*);
   ~Paddle();

//Implementation
public:
   virtual void move();
   virtual void move(CPoint& point);
    void initialize(CRect&);
   void newBoundingRectangle();


protected:
   DECLARE_DYNCREATE(Paddle)
private:
   const int_horizontalPaddleOffset;
   const int_verticalPaddleOffset;
   const int _moveOffset;
   const int   _bottomOffset;
};

#endif
```

PADDLE.CPP

```
//////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.           /
// Copyright (C) 1996 Software Architects,       /
// All rights reserved.                          /
//                                               /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                   /
//////////////////////////////////////////////////////////
// paddle.cpp: Defines the class behaviors for the paddle.
//
//
#include "stdafx.h"
#include "paddle.h"
#include "resource.h"
#include "puck.h"
#include "plyngfld.h"


/////////////////////////////////////////////////////////////////////////////
// Paddle
IMPLEMENT_DYNCREATE(Paddle, CObject)

Paddle::Paddle ():
   _horizontalPaddleOffset(30),
   _verticalPaddleOffset(10),
   _bottomOffset(40),
   _moveOffset(50)
{
   _objBitmap->LoadBitmap(PADDLEBITMAP);
   this->newBoundingRectangle();
}

Paddle::Paddle(PlayingField* fieldPtr):
   _horizontalPaddleOffset(30),
```

```
   _verticalPaddleOffset(10),
   _bottomOffset(40),
   _moveOffset(50)
{

   _objBitmap = new CBitmap;
   _fieldPtr = fieldPtr;
   _objBitmap->LoadBitmap(PADDLEBITMAP);
   this->newBoundingRectangle();
}

Paddle::~Paddle(){}

void Paddle::newBoundingRectangle(){

   _boundingRect.left = _posField.x - _horizontalPaddleOffset;
   _boundingRect.top = _posField.y - _verticalPaddleOffset;
   _boundingRect.right = _posField.x + _horizontalPaddleOffset;
   _boundingRect.bottom = _posField.y + _verticalPaddleOffset;
}


void Paddle::move(){
/* CSize off-
         set((int)_currentVelocity.getSpeedX(),(int)_currentVelocity.getSpeedY
         ());
   CRect rect = _fieldPtr->getBoundingRect();
   CPoint tempPtr(_posField+offset);
   this->setPosition(tempPtr);
   this->newBoundingRectangle();
   if (tempPtr.x > rect.right - _horizontalPaddleOffset){
      tempPtr.x = rect.right - _horizontalPaddleOffset;
      this->setPosition(tempPtr);
      _boundingRect.left = rect.right - _moveOffset;
      _boundingRect.right = rect.right;
   }
   else {
      _boundingRect.left = tempPtr.x;
      _boundingRect.right = tempPtr.x + _moveOffset;
   }
*/
}


void Paddle::move (CPoint& Point)
{
   CRect rect = _fieldPtr->getBoundingRect();
   this->_posField.x = Point.x;
   this->_posField.y = rect.bottom - 40;
   this->newBoundingRectangle();
   if (Point.x >= rect.right - _moveOffset){
      _boundingRect.left = rect.right - _moveOffset;
      _boundingRect.right = rect.right;
   }
   else {
      _boundingRect.left = Point.x;
      _boundingRect.right = Point.x + _moveOffset;
   }
}
```


PLYNGFLD.H

```
////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                    /
// Copyright (C) 1996 Software Architects,                /
// All rights reserved.                                   /
//                                                        /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                            /
////////////////////////////////////////////////////////////
// PlyngFld.h

#ifndef PLYNGFLD_H
#define PLYNGFLD_H
```

```
/*
   A playing field is a two-dimensional area in which an arcade
   game is played.  The area comprises a finite set of discrete
   locations, addressed using Cartesian coordinates.  By default,
   a playing field is rectangular in shape, bounded by four walls.

   A playing field is responsible for coordinating the interactions
   among various arcade game pieces.  Among its actions is the
   detection of collisions between pieces.  It does not process
   those collisions, instead rejecting any change that occurs in
   a piece by throwing a "collision" exception.
*/

#include "ArcGmObj.h"

class CPoint;
class ArcadeGamePiece;
class MovablePiece;
class Puck;
class CBricklesDoc;

class PlayingField : public ArcadeGameObject {
   public:
      PlayingField();//CBricklesDoc*);
      ~PlayingField();

      void tick();

   protected:
      CBricklesDoc* _theDoc;

   private:

};

#endif
```

PLYNGFLD.CPP

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                     /
// Copyright (C) 1996 Software Architects,                 /
// All rights reserved.                                    /
//                                                         /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                             /
/////////////////////////////////////////////////////////////
#include "stdafx.h"
//#include "puck.h"
#include "plyngfld.h"
//#include "brickdoc.h"

PlayingField::PlayingField()//CBricklesDoc* theDoc)
{
//  _theDoc = theDoc;
}

PlayingField::~PlayingField()
{

}

void PlayingField::tick()
{
//  _boundingRect = _theDoc->getBoundingRect();
}
```

PUCK.H

```
/////////////////////////////////////////////////////////////
```

```
// Brickles --  C++ Case Study Course.                    /
// Copyright (C) 1996 Software Architects,                /
// All rights reserved.                                   /
//                                                        /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                            /
//////////////////////////////////////////////////////////
// puck.h : Declares the class interfaces for the Bounce child window.
//


#ifndef __PUCK_H__
#define __PUCK_H__

/////////////////////////////////////////////////////////////////////////

class MovablePiece;
class PlayingField;
class Paddle;
//class Velocity;

#include "mvblpc.h"

class Puck : public MovablePiece
{
public:
   Puck();
   Puck(PlayingField*);
   ~Puck();

// Implementation
public:
   void newBoundingRectangle();
// virtual void initialize();
   virtual void setPosition(CPoint newCenter);
   virtual void move();
   virtual void move(CPoint&){}


protected:
   DECLARE_DYNCREATE(Puck)


protected:

   const int _puckRadius;
};

/////////////////////////////////////////////////////////////////////////

#endif // __PUCK_H__
```

PUCK.CPP

```
//////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                    /
// Copyright (C) 1996 Software Architects,                /
// All rights reserved.                                   /
//                                                        /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                            /
//////////////////////////////////////////////////////////
// puck.cpp: Defines the class behaviors for the puck.
//
//
#include "stdafx.h"
#include "puck.h"
#include "resource.h"
#include "plyngfld.h"
//#include "math.h"
#include "brickvw.h"

/////////////////////////////////////////////////////////////////////////
// Puck
IMPLEMENT_DYNCREATE(Puck, CObject)
```

```
////////////////////////////////////////////////////////////////////////////////////
// Puck creation


Puck::Puck():MovablePiece(0,CPoint(100,100), 4,50),
   _puckRadius(10)
{
   _objBitmap = new CBitmap;
   _objBitmap->LoadBitmap(PUCKBITMAP);
   this->newBoundingRectangle();

}

Puck::Puck(PlayingField* fieldPtr):
   MovablePiece(fieldPtr,CPoint(300,200), 10,50),
   _puckRadius(10)
{
   _objBitmap = new CBitmap;
   _objBitmap->LoadBitmap(PUCKBITMAP);
   this->newBoundingRectangle();
}

//void Puck::initialize()
//{
// this->newBoundingRectangle();
//
//}

Puck::~Puck(){

}

void Puck::newBoundingRectangle(){

   _boundingRect.left = _posField.x - _puckRadius;
   _boundingRect.right = _posField.x + _puckRadius;
   _boundingRect.top = _posField.y - _puckRadius;
   _boundingRect.bottom = _posField.y + _puckRadius;

}


void Puck::setPosition(CPoint newCenter)
{
   ArcadeGamePiece::setPosition(newCenter);
   this->newBoundingRectangle();
}


// Animate the puck.

void Puck::move(){
   CSize off-
         set((int)_currentVelocity.getSpeedX(),(int)_currentVelocity.getSpeedY
         ());
   CPoint tempPtr(_posField+offset);
   this->setPosition(tempPtr);
   this->newBoundingRectangle();
   CRect tmpRect = _fieldPtr->getBoundingRect();
   if(_leadingPoint.x < 0){
      _currentVelocity.reverseX();
      _posField.x=0;
   }
   else if(_leadingPoint.x > tmpRect.right){
      _currentVelocity.reverseX();
      _posField.x = tmpRect.right;
   }
   else if(_leadingPoint.y < 0){
      _currentVelocity.reverseY();
      _posField.y=0;
   }
   else if(_leadingPoint.y > tmpRect.bottom){
      _currentVelocity.reverseY();
      _posField.y = tmpRect.bottom;
   }
   _leadingPoint = getLeadingPoint();
}
```

RESOURCE.H

```
//{{NO_DEPENDENCIES}}
// App Studio generated include file.
// Used by BRICKLES.RC
//
#define IDR_MAINFRAME                2
#define IDD_ABOUTBOX                 100
#define IDB_BITMAP1                  103
#define PUCKBITMAP                   103
#define IDD_PLAYBOX                  104
#define PADDLEBITMAP                 106

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS

#define _APS_NEXT_RESOURCE_VALUE        107
#define _APS_NEXT_COMMAND_VALUE         32771
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

STDAFX.H

```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions (including VB)
```

STDAFX.CPP

```
// stdafx.cpp : source file that includes just the standard includes
//  stdafx.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

TIMER.H

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                      /
// Copyright (C) 1996 Software Architects,                  /
// All rights reserved.                                     /
//                                                          /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                              /
/////////////////////////////////////////////////////////////

#ifndef TIMER_H
#define TIMER_H

/*
   A timer is a device that alerts up to MAX arcade game objects to
   the passage of a unit interval of time.  An arcage game object can
   be enrolled with a timer.  When a unit time interval passes, each
   enrolled object is sent a "tick()" message.  An arcade game object
   can be withdrawn from the enrollment list for a timer.

   A timer can be disabled to inhibit notification of all enrollees.
*/

#include "stdafx.h"


class ArcadeGameObject;
```

```
class CWnd;

//template<int max, int unitInterval>
class Timer {
  public:
    Timer();
    Timer(CWnd*,int,int);// double check parameter and type
    ~Timer();

    void enroll(ArcadeGameObject*)const;// throw(OverFlow);
    void withdraw(ArcadeGameObject*)const;
    void enable();
    void disable();
    void tick();

    BOOL isEnabled() const;

  protected:

  private:
    int _enrolleeCount;
    ArcadeGameObject* _enrollees[10];
    int _whichTimer;
    CWnd* _whichWindow;
    BOOL _enabled;
    int max;
    int unitInterval;
};

/*
Notes:

  1.  It would be nice if a regidtered arcade game object could specify
    the member function to call back instead of standardizing on "tick".  For
    example, a puck might want a "move()" message instead of "tick()."  Page
        167
    of Stroustrup shows how to use pointers to member functions, but it is
    not clear that dynamic binding will take place on a message send--or even
    if sending that message to a subclass instance will be accepted.  So this
    will probably remain a nicety!

  2.  We might want to make this a singleton.

  3. No virtual methods!
*/

#endif
```

TIMER.CPP

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                        /
// Copyright (C) 1996 Software Architects,                    /
// All rights reserved.                                       /
//                                                            /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                                /
/////////////////////////////////////////////////////////////
// Timer.cpp

#include "stdafx.h"
#include "timer.h"
#include "arcgmobj.h"


    //template<int max, int unitInterval>
    Timer/*<max,unitInterval>*/::Timer(CWnd* window,int maxMembers,int
        interval){// double check parameter and type
      _whichWindow = window;
      _enrolleeCount = 0;
      _whichTimer = 1;
      max = maxMembers;
      unitInterval = interval;

    }

    Timer::Timer(){
```

```
   _enrolleeCount = 0;
   _whichTimer = 1;
   max = 5;
   unitInterval = 5;
}

//template<int max, int unitInterval>
Timer/*<max,unitInterval>*/::~Timer(){
}

//template<int max, int unitInterval>
void Timer/*<max,unitInterval>*/::enroll(ArcadeGameObject* item)const{
   if (_enrolleeCount <= max){
      _enrollees[_enrolleeCount]= item;
      ((Timer*)this)->_enrolleeCount = ((Timer*)this)->_enrolleeCount + 1;
   }

}

//template<int max, int unitInterval>
void Timer/*<max,unitInterval>*/::withdraw(ArcadeGameObject*
      item)const{
   int i = 0;
   while (i < _enrolleeCount && _enrolleeCount > 0){
      if (item = _enrollees[i]){
         if (i < _enrolleeCount-1){
            _enrollees[i] = _enrollees[_enrolleeCount];
         }
         ((Timer*)this)->_enrolleeCount = ((Timer*)this)->_enrolleeCount -
      1;
      }
   }
}

 //template<int max, int unitInterval>
void Timer/*<max,unitInterval>*/::tick(){
   for(int i=0;i<_enrolleeCount; i++){
      _enrollees[i]->tick();
   }
}

//template<int max, int unitInterval>
void Timer/*<max,unitInterval>*/::enable(){
   _whichWindow->SetTimer(_whichTimer,unitInterval,0);
}

//template<int max, int unitInterval>
void Timer/*<max,unitInterval>*/::disable(){
   _whichWindow->KillTimer(_whichTimer);
}

//template<int max, int unitInterval>
BOOL Timer/*<max,unitInterval>*/::isEnabled() const{
   return _enabled;
}
```

VELOCITY.H

```
//velocity.h



/*

   A velocity is a change in position made by an object

   over a unit interval of time, characterized by a speed and

   a direction in some Cartesian coordinate system.  A velocity

   is a vector, having x and y components in the coordinate system.

   Direction is specified in degrees.

*/
```

```
#ifndef VELOCITYH

#define VELOCITYH


typedef int Speed;

typedef int Direction;


class Velocity{

   public:

      Velocity();

      Velocity(const Speed, const Direction);

      Velocity(Velocity&);


      Speed getSpeed()const;

      Speed getSpeedX()const;

      Speed getSpeedY()const;

      Direction getDirection()const;

      void setSpeed(const Speed);

      void setDirection(const Direction);

      void reverseY();

      void reverseX();


   protected:

      int _dx;

      int _dy;

      int _speed;

      int _direction;


};


#endif
```

VELOCITY.CPP

```
/////////////////////////////////////////////////////////////
// Brickles --  C++ Case Study Course.                    /
// Copyright (C) 1996 Software Architects,                 /
// All rights reserved.                                  /
//                                                       /
// This source code is only intended as a supplement to the /
// C++ Case Study Course Notes                            /
/////////////////////////////////////////////////////////////
//velocity.cpp
```

```
#include “stdafx.h”

#include “velocity.h”

#include “math.h”


    Velocity::Velocity():_dx(0),
      _dy(0),
      _direction(0),
      _speed(0)
    {
    }


    Velocity::Velocity(const Speed newSpeed, const Direction newDirection):
      _dx((int)(cos((double)newDirection/57.2977)*newSpeed)),
      _dy((int)(sin((double)newDirection/57.2977)*newSpeed)),
      _direction(newDirection),
      _speed(newSpeed)
    {

    }
```

```
Velocity::Velocity(Velocity& oldVelocity){
  _dx = oldVelocity._dx;
  _dy = oldVelocity._dy;
}


Speed Velocity::getSpeed()const{
return _speed;
}


Speed Velocity::getSpeedX()const{
return _dx;
}


Speed Velocity::getSpeedY()const{
return _dy;
}


Direction Velocity::getDirection()const{
return _direction;
}
```

```
void Velocity::setSpeed(const Speed newSpeed){
_speed = newSpeed;
_dx = (int)(cos((double)_direction/57.2977)*_speed);
_dy = (int)(sin((double)_direction/57.2977)*_speed);
}

void Velocity::setDirection(const Direction newDirection){
_direction = newDirection;
_dx = (int)(cos((double)_direction/57.2977)*_speed);
_dy = (int)(sin((double)_direction/57.2977)*_speed);
}

void Velocity::reverseX(){
_dx = -_dx;
//	_direction = (int)(
//	   acos((double)(_dx) / (double)(_speed))
//	      * (360.0 / (2.0 * 3.1459265)) );
   if ( _direction < 180 ) {
      _direction = 180 - _direction;
   }
   else {
      _direction =  540 - _direction;
   };
}

void Velocity::reverseY(){
_dy = -_dy;
//	_direction = (int)(
//	   asin((double)(_dy) / (double)(_speed))
//	      * (360.0 / (2.0 * 3.1459265)) );
   _direction = (360 - _direction) % 360;
}
```