BoardSpace

A Place for Board Games

Games at Boardspace.net

Euphoria	Hive	Tammany Hall	SixMaking	9 Men Morris	Stac	Checkers	Proteus	Majorities
Carnac	PonteDelDiavolo	Morelli	Medina	Gounki	Gyges	Kamisado	Zèrtz	Yinsh
Dvonn	Tzaar	Gipf	Go	Shogi	Xiangqi	Tablut	Rithmomachy	Fanorona
Pünct	1 Day in London	Yspahan	Raj	mogul	Container	Army of Frogs	Che	Micropul
Palago	Spangles	Trax	Santorini	Exxit	Gobblet	Lines of Action	Plateau	Hex
Tumbling Down	Dipole	Truchet	Volcano	Traboulet	Qyshinsu	Knockabout	Warp6	Breaking Away
Cannon	Triad	Mutton	Octiles	Colorito	Arimaa	Crossfire	Entrapment	Quinamid

Boardspace provides

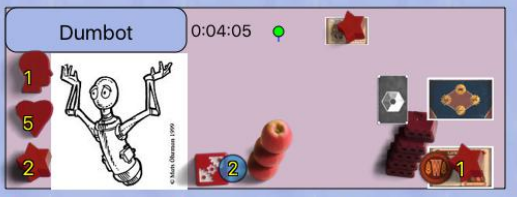
Live play over the internet of board games for 2-6 players

- Restartable
- Archived and searchable and replayable games
- Player Rankings
- Optional robot opponents


Review Games #3
Review Games #3 Chat
game selector
null : Game won by null
Tile Size
null
0:00:00
bS2
wS1
bG2
bS1
L1
bG1
bB2
bB1
wB2
wG2
bM1
wQ
bL1
wM1
wS2
bA1
bA2
52
Done
Edit
null
0:00:00
surround your opponents queen bee

MrBrains : Game won by MrBrains

94 MrBrains to Generator
95 Dumbot to Build Market A
Pay with , gain #1Morale (from Flartner the Luddite)
96 MrBrains Retrieve from Worker Activation B
96 MrBrains Retrieve from Generator -1
Green lose 1 or 1 due to roll penalties
97 Dumbot to Tunnel
98 MrBrains to Generator
Extra energy cancelled by Registry of Personal Secrets
99 Dumbot to Build Market B
Pay with , gain #1Morale (from Flartner the Luddite)
100 MrBrains Retrieve from Generator
101 Dumbot Retrieve from Tunnel

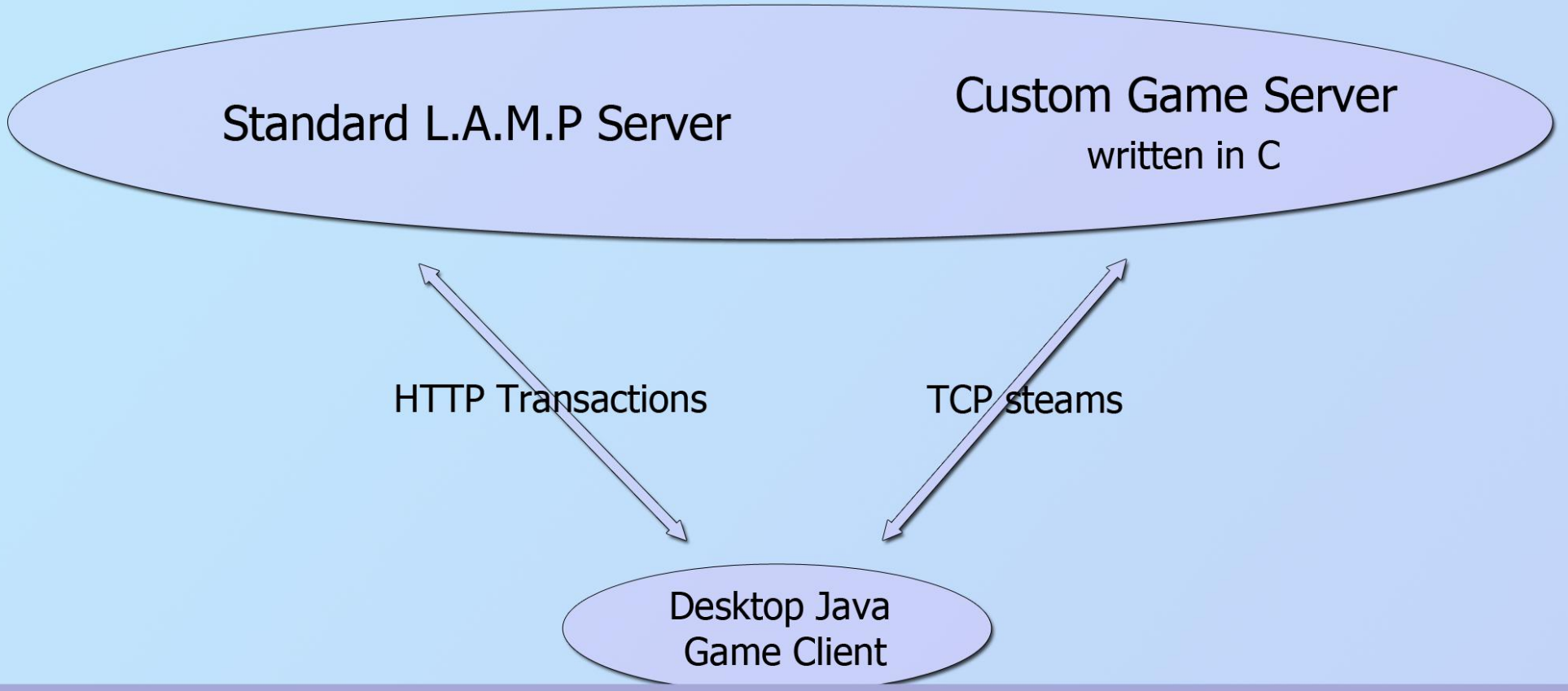


Done | Edit

Lobby: Welcome to BoardSpace client, version 3.1
News:
News: Remember that you are a spectator.
News: Please do not comment on the game.
News:
Game: Played on Sun Mar 13 13:40:06 CDT 2016
prop RE = Game won by MrBrains

To ALL: | Type your message here. | Send

Boardspace.net Architecture



Lots of Java Code

50,000 lines core code + 4,000 lines per game x 75 games

Motivation for Change

Declining Browser based Java Support

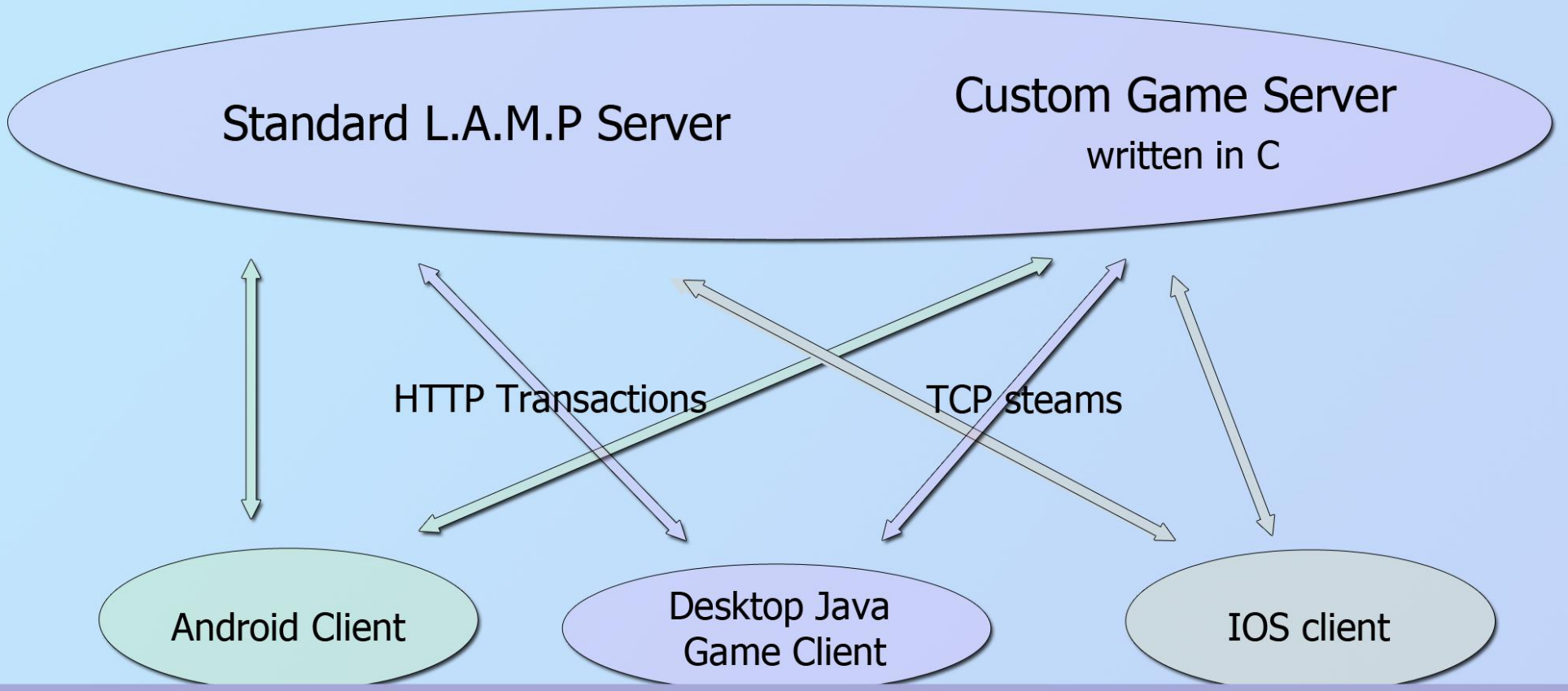
- java exploit/patch wars
- dropped Applet support on chrome
- no Applets on Edge.
- pending end of Applets on firefox.

Increasing Mobiles mindshare

- android/ios phones everywhere
- android tablets are cheap and plentiful

Proposed new

Boardspace.net Architecture



Lots of Java Code

50,000 lines core code + 4,000 lines per game x 75 games

Possible Porting Aproaches

Native Code

- Android: Still Java but vastly different UI, event and networking
- Android: Morass of version and device specifics

- IOS: complete rewite, no common code

Robovm

- IOS: Closed source, newish
- and as of 4/2016, bought and killed by Microsoft.

Codenameone

- Open source, multi-target Android, IOS and others.
- compatible UI, Event and networking

How Codenameone works

Eclipse / Netbeans Plugin
Embedded Simulator

Codename1 API,
- similar to the standard java desktop environment

The “build” process recodes your JAR files to native code for the target platform, compiles using native tools, and packages the result for delivery.

Delivered source code is available.

Porting Tasks

Desktop + Android + Ios apps

Many cosmetic Java code changes

A Few mandatory new features

7000 lines of new Java

Minor tweaks to HTTP/Lamp code

No changes to TCP networking

Shared code base for Desktop, Android and Ios

6 Months of work

Cosmetic Changes: Rectangle

Standard Java

```
public class Rectangle
{ public int x,y,width,height;
  public double getX();
  public double getY();
  public double getWidth();
  public double getHeight();
}
```

Codenameone

```
public class Rectangle
{  private int x,y,width, height;
   public int getX();
   public int getY();
   public int getWidth();
   public int getHeight();
}
```

Functionally equivalent, operationally completely incompatible.

Standard Java Rectangle

```
boardRect.x = 0;
boardRect.y = (wideMode ? 0 : chatHeight)+SQUARESIZE-CELLSIZE;
boardRect.width = SQUARESIZE * b.boardColumns ;
boardRect.height = SQUARESIZE * b.boardRows;


firstPlayerChipRect.x = G.Right(boardRect)-CELLSIZE/2;
firstPlayerChipRect.y = boardRect.y+CELLSIZE/2;
firstPlayerChipRect.width = SQUARESIZE;
firstPlayerChipRect.height = SQUARESIZE;
```

Codenameone Rectangle

```
G.SetRect(boardRect,boardX,boardY,boardW,boardH);

G.SetRect(firstPlayerChipRect,
   G.Left(boardRect),
    G.Top(boardRect),
   SQUARESIZE,
    SQUARESIZE);
```

Mandatory new Features

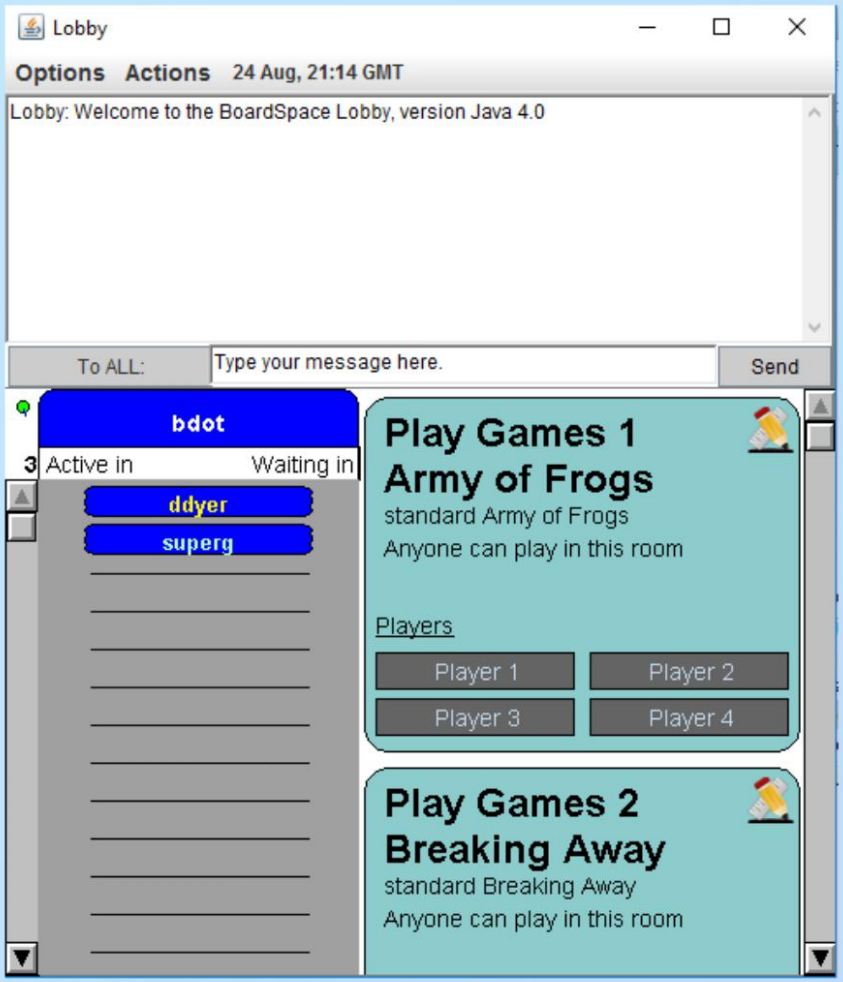
One Finger Scrolling
Two finger “pinch” zoom.
Cope with the absence of mouse rollovers

Scale any remaining Fixed size UI
Cope with fixed aspect ratios
Tabbed full screen windows.

Scale robots for CPU performance

Cope with small physical screen sizes

Boardspace Lobby



Uses no java widgets except in the chat area

Has

Left and Right Scroll Bars

Fixed layout proportions for chat, player list, game room list.

Fixed layout within game rooms.

Needs

Finger "fling" scrolling

Flexible proportions to major parts

Scaled boxes within game rooms

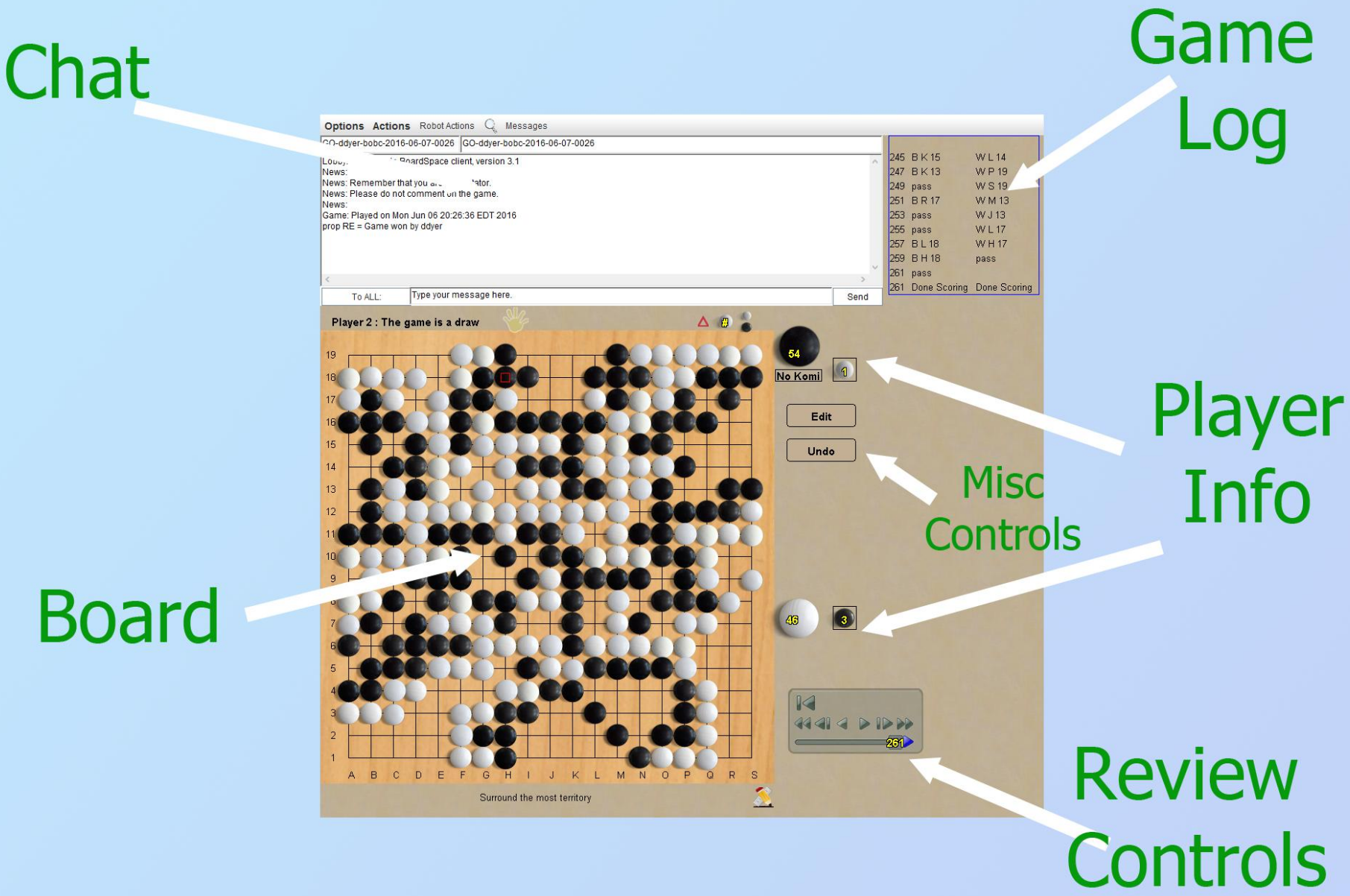
Boardspace Lobby

Horizontal Phone Screen

Ios Screen

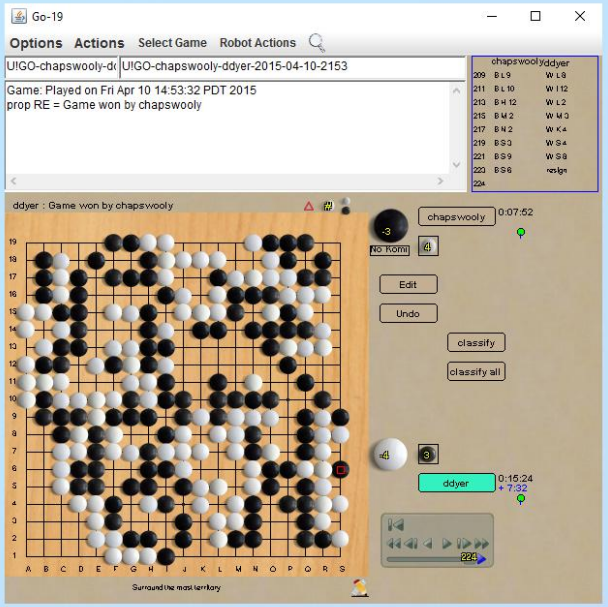
Vertical Phone Screen

Boardspace typical game room

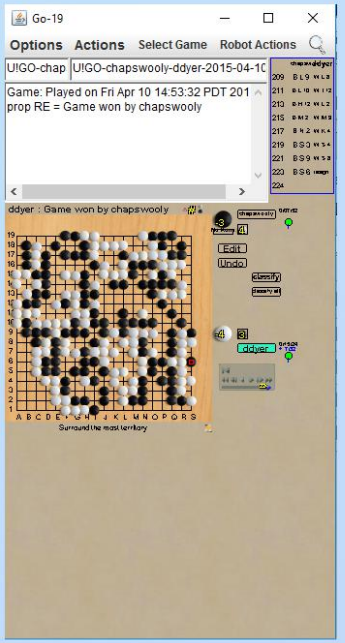


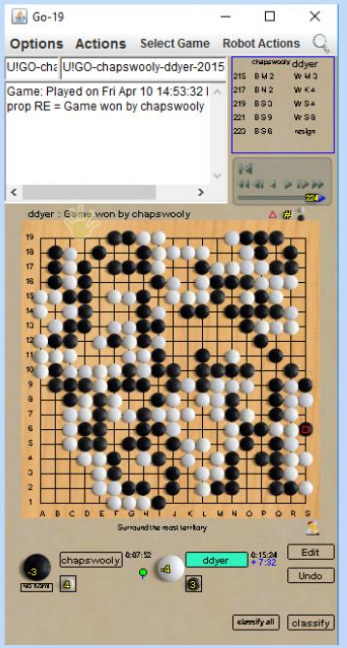
Fixed window sizes, variable aspect ratio

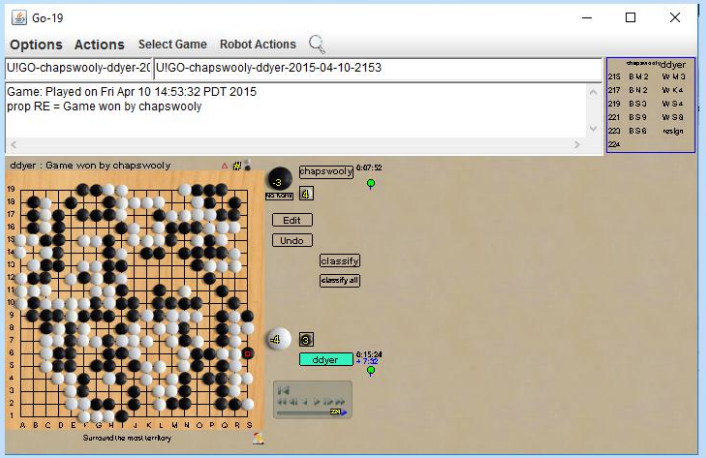
Normal



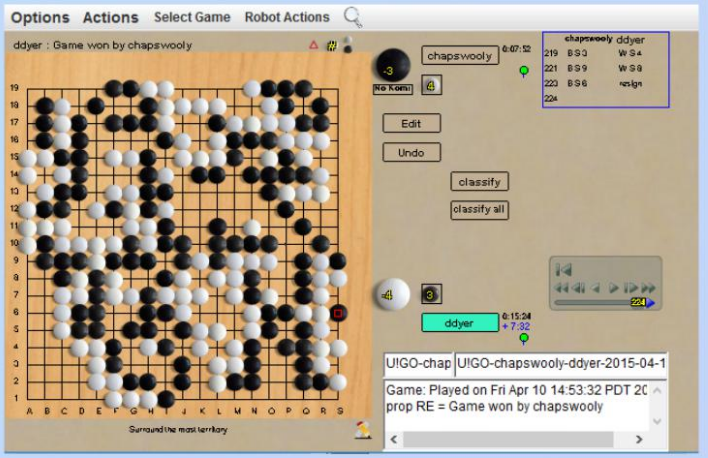
Tall







Wide

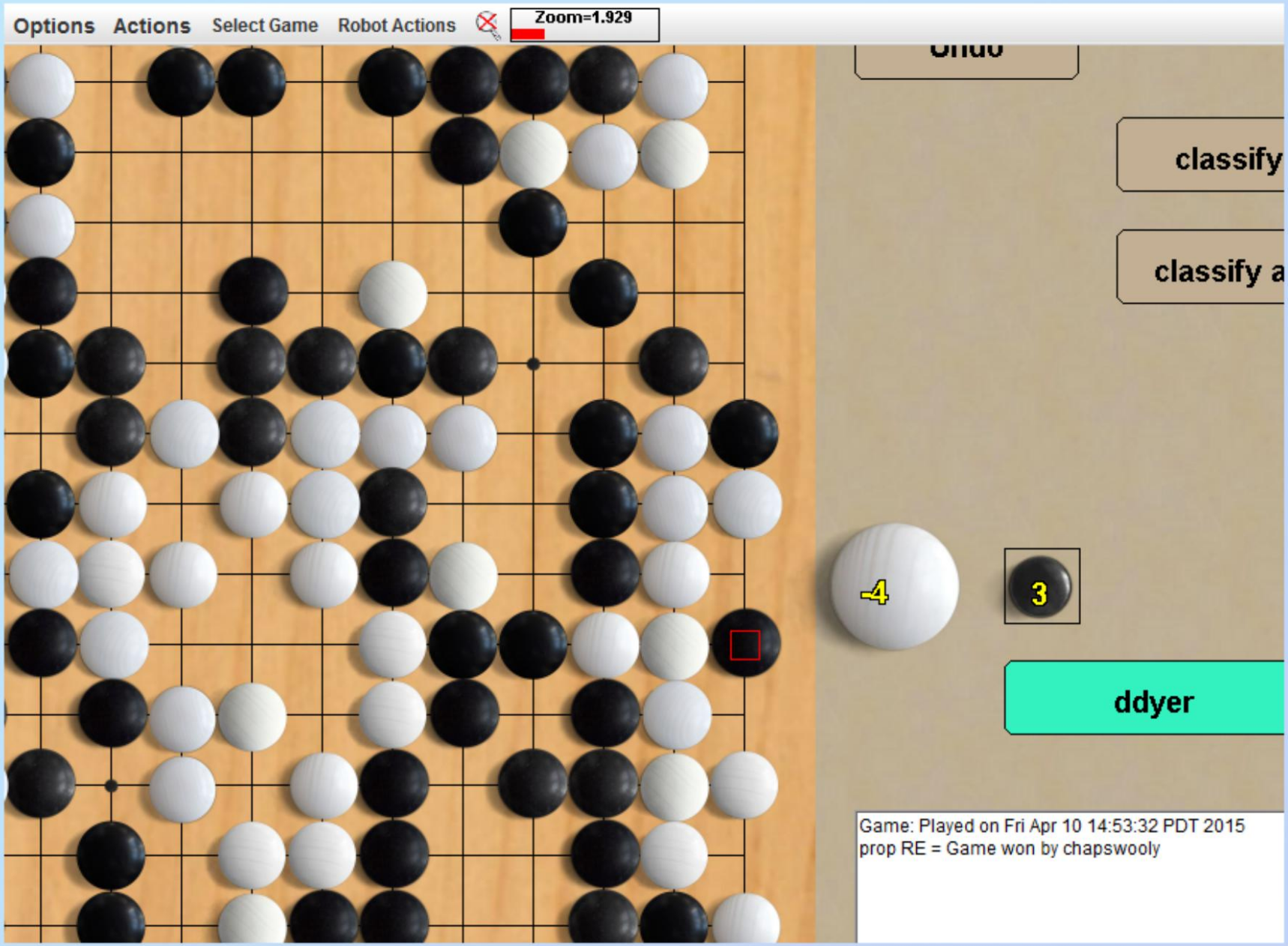


Typical Layout Code

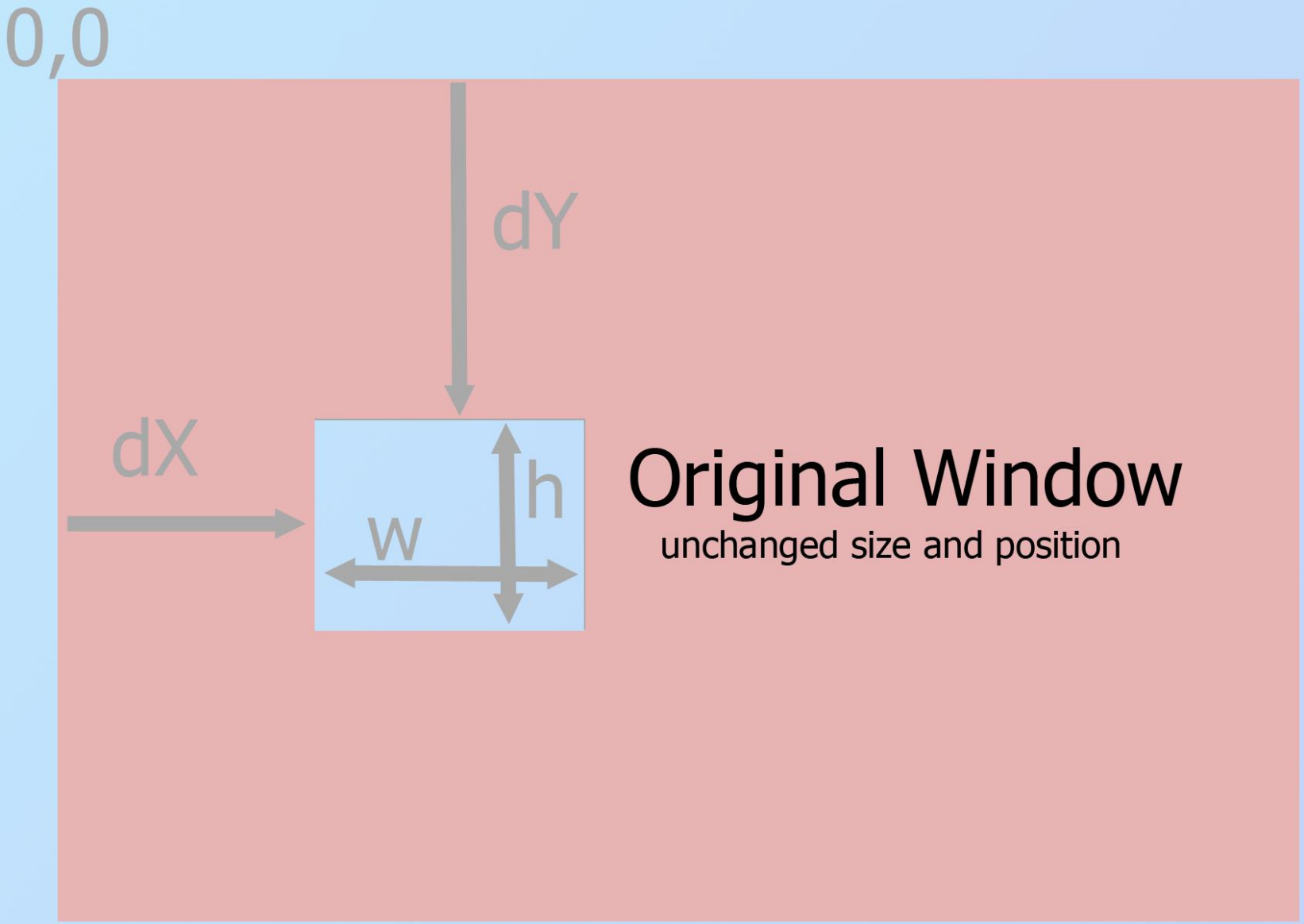
```
public void setLocalBounds(int x, int y, int width, int height)
{   int normal = setLocalBoundsCell(width,height,false,false);
  boolean wide = setLocalBoundsCell(width,height,true,false)>normal;
  boolean tall = setLocalBoundsCell(width,height,false,true)>normal;

  setLocalBoundsWT(x,y,width,height,wide,tall);
}
int setLocalBoundsCell(int width,int height,boolean wideMode,boolean tallMode)
{   int sncols = (b.boardColumns+(tallMode ? 1 : (wideMode ? 13 : 9))); // more cells wide to allow for the aux displays
    int snrows = (b.boardRows+1)+(tallMode ? 6 : 0);
    double cellw = (double)width / sncols;
    int chatHeight = selectChatHeight(height);
    double cellh = (double)(height-(wideMode ? 0 : chatHeight)) / snrows;
    double cs = Math.max(1,Math.min(cellw, cellh)); //cell size appropriate for the aspect ration of the canvas
    int CSIZE = (int)(cs*b.boardColumns/40);
    CELLSIZE = (int)cs;
    return(CSIZE);
}
void setLocalBoundsWT(int x,int y,int width,int height,boolean wideMode,boolean tallMode)
{ int CSIZE = setLocalBoundsCell(width,height,wideMode,tallMode);
  int ideal_logwidth = CSIZE * 13;
    int chatHeight = selectChatHeight(height);
    G.SetRect(fullRect,0,0,width,height);
    int boardW = CELLSIZE * (b.boardColumns-1);
    G.SetRect(boardRect,tallMode ? x+(width-boardW)/2 : x,wideMode ? CSIZE : chatHeight+CSIZE,
     boardW,CELLSIZE * b.boardRows);


    // game log.  This is generally off to the right, and it's ok if it's not
    // completely visible in all configurations.
    G.SetRect(stateRect,G.Left(boardRect) + CSIZE,(wideMode ? 0 : chatHeight) +CSIZE/3,
        G.Width(boardRect) - CSIZE,CSIZE*2);
```

Pinch Zoom

mouse zoom on desktops



Zoom low level Layout



Zoom Code

```
public void actualPaint(Graphics g,HitPoint hp)
{
    int x = getSX();
    int y = getSY();
    //Rectangle clip = G.setClip(g,0,0,getWidth(),getHeight());
    super.paint(g);
    g.translate(-x,-y);
    paintSprites(g,hp);
    //G.setClip(g,clip);
    g.translate(x,y);
}
```

- The real window system doesn't even know your doing this
- Depends on actual clipping logic to be reasonably efficient
- The tricky bit is getting the math right to set the window.

Window Size and Position

Free form, multiple windows

Tabbed fixed size windows

New Feature: CPU benchmark

```
private static int fact(int n)
{
    return(n==0 ? 1 : n*fact(n-1));
}

private static double speed = 0.0;

public static double cpuTest()
{ long now = G.Date();
  for(int j=0;j<1000000;j++) { fact(20); }
   long later = G.Date();
  // returns 1.0 on my desktop on 1/2016
  return(28.80/(later-now+1));
}
```

Apply speed test to all robots, disallow using robots to slow for your platform.

Add new robots that use less resources.

New Feature: Feature size test

```
/**
 * return the smallest recommended feature size, in pixels.  This is
 * to be used to configure touch screens, where fat fingers can't point
 * at anything too small.
 * @return smallest recommended feature size, in pixels
 */
public static int minimumFeatureSize()
{
  return(isTouchInterface() ? getPPI()/3 : 10);
}
```

Apply to Review controls, potentially others.

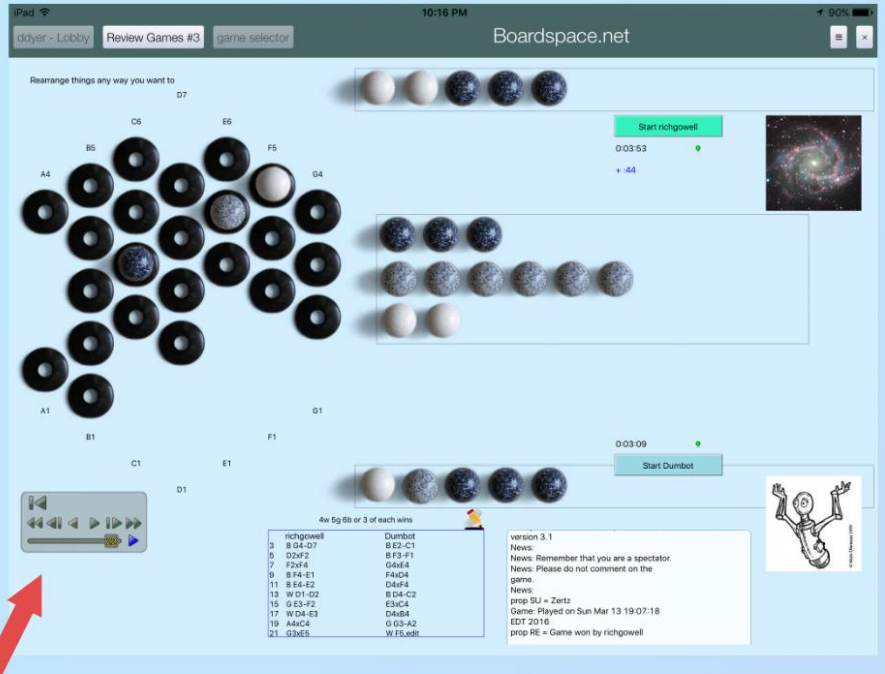
Typical Device Characteristics

cpu=30% screen=1680x1050 ppi=116 Mc OS X
cpu=10% screen=1680x1050 ppi=116 Mac OS X
cpu=30% screen=1680x1050 ppi=116 Mac OS X

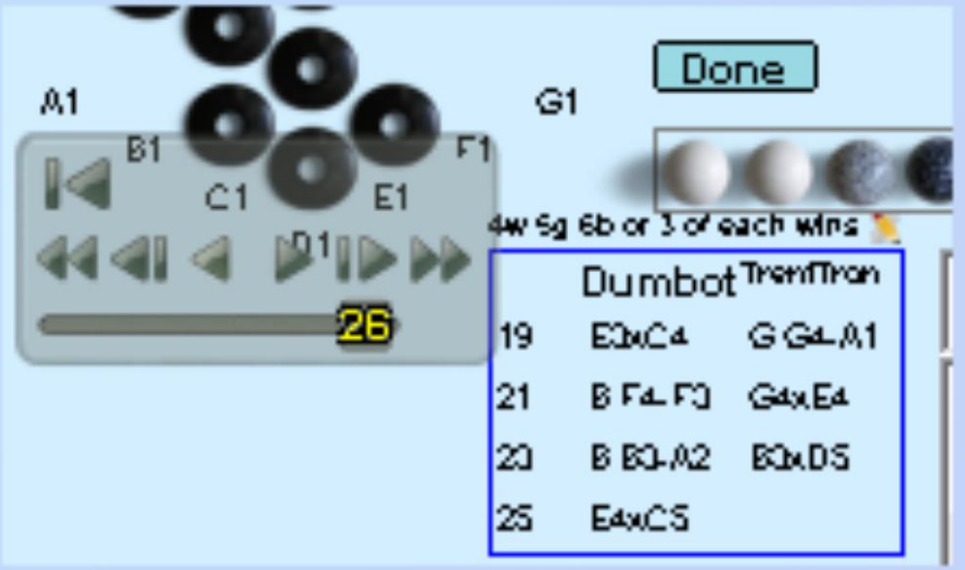
cpu=13% screen=1024x600 ppi=96 Windows 7
cu=92% screen=1920x1080 ppi=96 Windows 10
cpu=36% screen=1920x1080 ppi=96 Windows Vista
cpu=80% screen=1920x1200 ppi=96 Windows 7
cpu=46% screen=1024x768 ppi=96 Windows 10

cpu=12% screen=1200x1776 ppi=160 Codename1 1.59 Android
cpu=2.1% screen=1280x767 ppi=128 Codename1 1.59 Android
cpu=5.7% screen=768x1134 ppi=160 Codename1 1.59 Android
cpu=9.0% screen=1196x672 ppi=160 Codename1 1.59 Android
cpu=4.8% screen=480x764 ppi=128 Codename1 1.56 Android

cpu=4.3% screen=1536x2048 ppi=160 Codename1 1.58 Ios
cpu=1.6% screen=2048x1536 ppi=160 Codename1 1.58 Ios
cpu=4.7% screen=2048x1536 ppi=160 Codename1 1.58 Ios
cpu=0.8% screen=1024x768 ppi=96 Codename1 1.58 Ios



Expand controls based on feature size



Bridge Needed: new Socket Connection

```
/** public utility function to open a socket */
  public Socket getSocketConnection(String server,int sock)
  {  Socket myS = null;

      if((myS==null) && (sock!=PROXY_HTTP_PORT))
       { myS = new Socket(server, sock);
      }
       return(myS);
  }
```

Socket Bridge Code

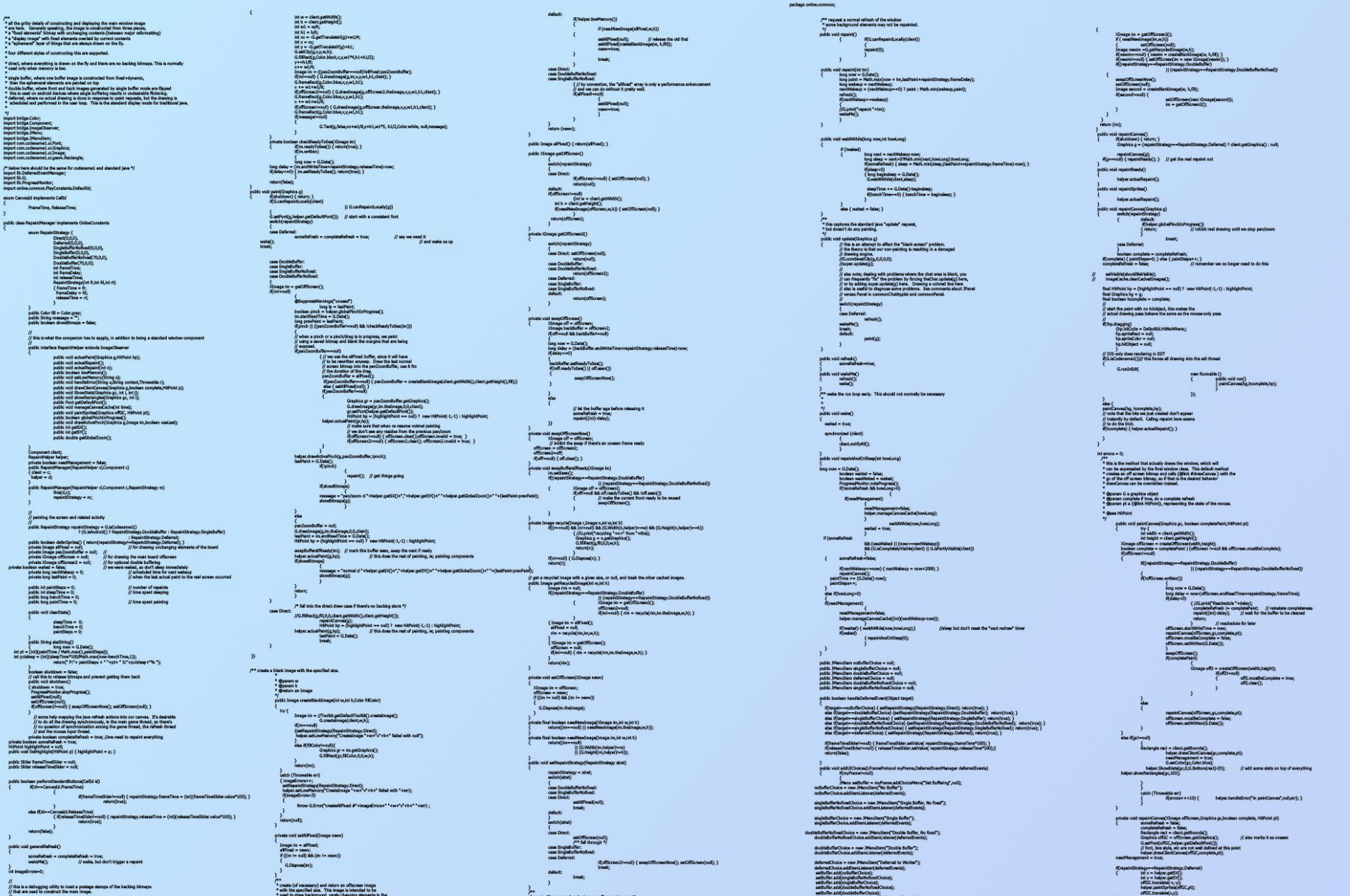
```
package bridge;

public class Socket extends SocketConnection // SocketConnection adds callback when the connection succeeds or fails
{
   // this is the API that standard java provides
   public Socket(String s,int n) throws IOException
   {  connect(s,n);
      if((input==null) || (output==null))
         { throw new IOException("Connect to "+s+":"+n+" got error "+errorCode+" - "+message);
         }}
   // do the connection and wait for the callback
   private synchronized void connect(String s,int n)
   {      ...
      com.codename1.io.Socket.connect(s,n,this);
      try {
         while (socketPending) { wait(); }
      }
      catch (InterruptedException err) {}
   }
   // on failure
   public synchronized void connectionError(int err, String errm)
   {  ...    }
   // on success
   public synchronized void connectionEstablished(InputStream is, OutputStream os)
   {  ...    }
   // a dummy method
   public String getHostAddress() { return("0.0.0.0"); }
   // not the usual definition, but only needed in transition to getHostAddress
   public Socket getLocalAddress() { return(this); }
}
```

Important point: The bridge socket implementation provides the functionality I need, not the full java socket API.

Trivial "MediaTracker" bridge
(codenameone doesn't use them)

```
package bridge;

import com.codename1.ui.Image;

public class MediaTracker {

    public MediaTracker(ImageObserver mapViewer) {
        // TODO Auto-generated constructor stub
    }

    public void addImage(Image map, int i) {
        // TODO Auto-generated method stub

    }

    public void waitForAll() throws InterruptedException {
        // TODO Auto-generated method stub

    }

    public void waitForID(int i) {
        // TODO Auto-generated method stub

    }

    public Object[] getErrorsAny() {
        return(null);
    }

}
```

Rather complex adaper for paint



Drawing Complications

Drawing everything every time is too slow.
 - poor animation frame rates and clunky object movement

Drawing with buffering glitches on Android

Drawing restricted to EDT thread

Solutions

Standard Java draws directly in user threads

IOS draws with single buffer in EDT thread

Android draws with double buffer in EDT thread

Development Environment

Eclipse or Netbeans, with plugin

Embedded Device Simulator

- Skins for common device formats
- Excellent replication of standard java functionality
- Good replication of display behavior
- No support for multi-touch screens.

On Device Debugging

- Direct installation of device builds
- Basically Nonexistent on-device debugging.
- Logging/error catching optional

Live Debugging tools

Do not expect bug reports!

Use try/catch in all processes and

Post Errors with HTTP

Log data as part of TCP stream (if active)

Use On Device Console Window for System.out

URLS

http://codenameone.com/

http://boardspace.net/

http://boardspace.net/english/codename1-port/Porting Boardspace.html

http://boardspace.net/english/codename1-port/Jug Presentation.pdf

ITunes, Google Play, Amazon Store: Boardspace.net