map

```
/* general planning
* baghdad on left, heian-kyo on right
* TODO:
* fix htdx so that it draws at correct location
*/



//declare images for maps
PImage bm; //baghdad map
PImage bt; //baghdad thumbnail
PImage hm; //heian-kyo map
PImage ht; //heian-kyo thumbnail

PGraphics bb; //buffers
PGraphics hb;

//declares buffers

//declares temporary render image variable
PImage tmpi;

//declare other information for maps
final int bmw = 737; //baghdad map width base
final int bmh = 590; //baghdad map height base
final int hmw = 430; //heian-kyo map width base
final int hmh = 590; //heian-kyo map height base
final int btp = 5; //baghdad thumbnail part
final int bmz = 3; //baghdad zoom
final int htp = 5; //heian-kyo thumbnail part
final int hmz = 3; //heian-kyo zoom
final int ww = bmw + hmw; //window width
final int wh = bmh; //window height. arbitrarily based off of baghdad.
final float sdl = 250; //lock/unlock delay
final int tl = 21; //table length
final int tlb = 9; //table last baghdad info, zero indexed
```

```
final float tbw = 120; //text box width for map labels
final float tbh = 100; //text box height for map labels
final float pi = 3.14159265358979323846264338327 95;

//declare information for mouse
float mx; //mouse x
float my; //mouse y
float bpx; //baghdad mouse percentage x
float bpy; //baghdad mouse percentage y
float hpx; //heian-kyo mouse percentage x
float hpy; //heian-kyo mouse percentage y

//declare variables for map rendering
float bdx; //baghdad map display x
float bdy; //baghdad map display y
float hdx; //heian-kyo map display x
float hdy; //heian-kyo map display y

//declare variables for thumbnail rendering
float btrx; //baghdad thumbnail rectangle x
float btry; //baghdad thumbnail rectangle y
float htrx; //heian-kyo thumbnail rectangle x
float htry; //heian-kyo thumbnail rectangle y

//declare temporary variables
float ta;

boolean bml;
boolean hml;

float blt;
float hlt;

float bmy;
float bmx;
float hmy;
float hmx;
```

```
//declare animation vars
float lto; //offset
float ltm; //millis
boolean ltb; //whether to not interrupt the other
boolean ltd; //true for heian, false for baghdad

//float[] pxs;
//float[] pys;
//String[] pts;
//String[] pss;

float[] pxs = new float[tl];
float[] pys = new float[tl];
String[] pts = new String[tl];
String[] pss = new String[tl];
float[] pus = new float[tl];
float[] pvs = new float[tl];


int btdx(float ta, float mxi) { //baghdad to display x, given x on unscaled map
 /* mx / bmw is percent of mouse to right
  * bmw - map.width is the leftmost point of the map as displayed
  * bmw is screen size, map.width is map size
  * + ta is added to the leftmost part to offset onto the map (add coordinate to map origin) also
including zoom
  */
 //println(ta, ta * bmz, mxi / bmw, bmw - bm.width);
 return int((mxi / bmw) * (bmw - bm.width) + ta * bmz);
}
int btdy(float ta, float myi) { //same as btdx, but for y
 return int((myi / bmh) * (bmh - bm.height) + ta * bmz);
}
int htdx(float ta, float mxi) { //heian-kyo to display x, given x on unscaled map
 return int(((mxi - bmw) / hmw) * (hmw - hm.width) + ta * hmz);
}
int htdy(float ta, float myi) { //same as htdx, but for y
```

```
 return int((myi / hmh) * (hmh - hm.height) + ta * hmz);
}



void render() {
 mx = mouseX; //gets mouse location
 my = mouseY;

 if (millis() - blt > sdl && keyPressed && key == &apos;b&apos;) {
  if (bml == false) {
   bml = true;
  } else {
   bml = false;
  }
  blt = millis();
 }
 if (millis() - hlt > sdl && keyPressed && key == &apos;h&apos;) {
  if (hml == false) {
   hml = true;
  } else {
   hml = false;
  }
  hlt = millis();
 }

 if (mx < bmw) { //makes sure that maps are locked if mouse is outside of correct area for
drawing
  hml = true;
 } else {
  bml = true;
 }
 if (!bml) { //if baghdad unlocked, move to cursor
  bmx = mx;
  bmy = my;
 }
 if (!hml) {/*!hml) { //if heian-kyo unlocked, move to cursor*/
```

```
  hmx = mx;
  hmy = my;
 }
 println(hmx, hmy);
 tint(255, 255); //sets to be solid and full color

 //draws baghdad map
 /* sets the temporary image to a section of the map
  * it gets the inverse of the converted coordinates (translation 0) so that they are positive
image coordinates instead of negative display coordinates
  * the width and height is the window size
  */
 bb.tint(255, 255);
 tmpi = bm.get(-1 * btdx(0, bmx), -1 * btdy(0, bmy), bmw, bmh); //gets baghdad map
 bb.image(tmpi, 0, 0); //draws temporary image that was retrieved earlier

 //draws thumbnail
 bb.tint(255, 127);
 bb.image(bt, 0, 0);
 bb.noFill();
 bb.rect(-1 * btdx(0, bmx) / btp / bmz, -1 * btdy(0, bmy) / btp / bmz, bmw / btp / bmz, bmh /
btp / bmz);


 //draws keian-kyo map
 //same as above, but with horizontal translation to get origin to match image origin
 hb.tint(255, 255);
 tmpi = hm.get(-1 * (htdx(0, hmx)), -1 * htdy(0, hmy), hmw, hmh); //gets heian-kyo map
 hb.image(tmpi, 0, 0); //draws image
 //println(hmx, hmy);

 //draws thumbnail
 hb.tint(255, 127);
 hb.image(ht, 0, 0);
 hb.noFill();
 hb.rect(-1 * (htdx(0, hmx)) / htp / hmz, -1 * htdy(0, hmy) / htp / hmz, hmw / htp / hmz, hmh /
htp / hmz);
```

```
 //draws mouse pointers
 bb.noFill();
 hb.noFill();
 bb.stroke(255, 0, 0);
 hb.stroke(255, 0, 0);
 bb.ellipse(bmx, bmy, 5, 5);
 hb.ellipse(hmx - bmw, hmy, 5, 5);

}

void settings() {
 size(ww, wh); //sets window dimensions
}

void setup() {
 bm = loadImage("bm.png"); //load baghdad map
 bt = loadImage("bm.png"); //load baghdad thumbnail
 hm = loadImage("nhmr.jpg"); //load heian-kyo map
 ht = loadImage("nhmr.jpg"); //load heian-kyo thumbnail

 bt.resize(bmw / btp, bmh / btp); //resize baghdad thumbnail to be fraction of original
 ht.resize(hmw / htp, hmh / htp); //resize heian-kyo thumbnail to be fraction of original
 bm.resize(bmw * bmz, bmh * bmz); //resize baghdad map to be zoomed
 hm.resize(hmw * hmz, hmh * hmz); //resize heian-kyo map to be zoomed

 bml = false;
 hml = false;

 blt = 0;
 hlt = 0;

 bb = createGraphics(bmw, bmh);
 hb = createGraphics(hmw, hmh);

 //table organization: x, y, tooltip, full text
```

```
 noFill(); //disables fill for thumbnail position indicators

 Table info = loadTable("info.tsv", "header"); //load table of text information, has a header

 //pxs = new float[tl];
 //pys = new float[tl];
 //pts = new String[tl];
 //pss = new String[tl];

 for (int i = 0; i < tl - 1; i++) {
   pxs[i] = info.getFloat(i, 0);
   pys[i] = info.getFloat(i, 1);
   pts[i] = info.getString(i, 2);
   pss[i] = info.getString(i, 3);
   pus[i] = info.getFloat(i, 4);
   pvs[i] = info.getFloat(i, 5);
 }
}



void bbuttons(float px, float py) { //px, py are offsets given to the function to display at the
right place

 for (int i = 0; i <= tlb; i++) {
   float x = pxs[i];
   float y = pys[i];
   String t = pts[i];
   String s = pss[i];
   bb.ellipse(px + x, py + y, 5, 5);
   bb.fill(0);
   bb.rect(px + x - 60, py + y + 10, 120, 100);
   bb.fill(255);
   bb.text(t, px + x - 60, py + y + 10, 120, 100);
   bb.noFill();
 }
```

```
}

void hbuttons(float px, float py) { //px, py are offsets given to the function to display at the
right place

 for (int i = tlb + 1; i < tl - 1; i++) {
   float x = pxs[i];
   float y = pys[i];
   String t = pts[i];
   String s = pss[i];
   hb.ellipse(px + x, py + y, 5, 5);
   hb.fill(0);
   hb.rect(px + x - tbw / 2, py + y + 10, tbw, tbh);
   hb.fill(255);
   hb.text(t, px + x - tbw / 2, py + y + 10, tbw, tbh);
   hb.noFill();
 }
}

void bdt(float mcx, float mcy) {
 float x = -1000;
 float y = -1000;
 int i = 0;
 String s = "";
 while (i <= tlb && (sqrt(pow((mcx - x), 2) + pow((mcy - y), 2)) >= 32)) {
   x = pxs[i];
   y = pys[i];
   s = pss[i];
   x = x + btdx(0, mcx);
   y = y + btdy(0, mcy);

   i++;
   //println(sqrt(pow((mcx - x), 2) + pow((mcy - y), 2)));
 }
 i--;
 //println(s);
 if (sqrt(pow((mcx - x), 2) + pow((mcy - y), 2)) < 32) {
```

```
  lto--;
  if (lto < -10) {
    lto = -10;
  }
  ltb = true;
  ltd = false;
  hmx = pus[i] + bmw;
  hmy = pvs[i];

 } else {
  if (lto < 0 && !ltb) {
    lto++;
  }
  ltb = false;
 }
 if (!ltd) {
  ltm = millis();
  fill(0);
  rect(0, bmh - tbh + an(10 * lto) + 100, bmw + hmw, tbh);
  fill(255);
  text(s, 0, bmh - tbh + an(10 * lto) + 100, bmw + hmw, tbh);
  noFill();
 }

}

void hdt(float mcx, float mcy) {
 float x = -1000;
 float y = -1000;
 int i = tlb + 1;
 String s = "";
 while (i < tl - 1 && (sqrt(pow((mcx - x), 2) + pow((mcy - y), 2)) >= 32)) {
  x = pxs[i];
  y = pys[i];
  s = pss[i];
  x = x + htdx(0, mcx + bmw);
  y = y + htdy(0, mcy);
```

```
  i++;
  //println(sqrt(pow((mcx - x), 2) + pow((mcy - y), 2)));
  //println(mcx - x, x, mcx, hmx, hmx - bmw, htdx(0, mcx));
 }
 i--;

 if (sqrt(pow((mcx - x), 2) + pow((mcy - y), 2)) < 32) {
  lto--;
  if (lto < -10) {
   lto = -10;
  }
  ltb = true;
  ltd = true;
  bmx = pus[i];
  bmy = pvs[i];
 } else {
  if (lto < 0 && !ltb) {
   lto++;
  }
  ltb = false;
 }
 if (ltd) {
  //println(s);
  fill(0);
  rect(0, bmh - tbh + an(10 * lto) + 100, bmw + hmw, tbh);
  fill(255);
  text(s, 0, bmh - tbh + an(10 * lto) + 100, bmw + hmw, tbh);
  noFill();
 }

}
float an(float in) {
 return sin(in / 200 * pi) * 100;
}

float am(float in) {
```

```
 return cos(in / 100 * pi) * 100;
}
void draw() {
 bb.beginDraw();
 hb.beginDraw();
 render();
 println(hmx, hmy);

 //bbuttons(btdx(0, bmx), btdy(0, bmy));
 bbuttons(btdx(0, bmx), btdy(0, bmy));
 hbuttons(htdx(0, hmx), htdy(0, hmy));

 bb.endDraw();
 hb.endDraw();
 image(bb, 0, 0);
 image(hb, bmw, 0);
 bdt(bmx, bmy);
 hdt(hmx - bmw, hmy);

}
```