

PROGRAMMERING C

Navn: Jacob Bom
Klasse: 15xar
Skole: Aarhus Gymnasium
Fag: Programmering C
Opgave: Miniprojekt
Vejleder: Mirsad Kadribasic
Aflevering: 8. februar 2017

Indholdsfortegnelse

Abstract

Denne rapport er opgave besvarelse på et miniprojekt som del af eksamensforberedelse i programmering C. Programmet som rapporten dækker over, er kodet i Processing.

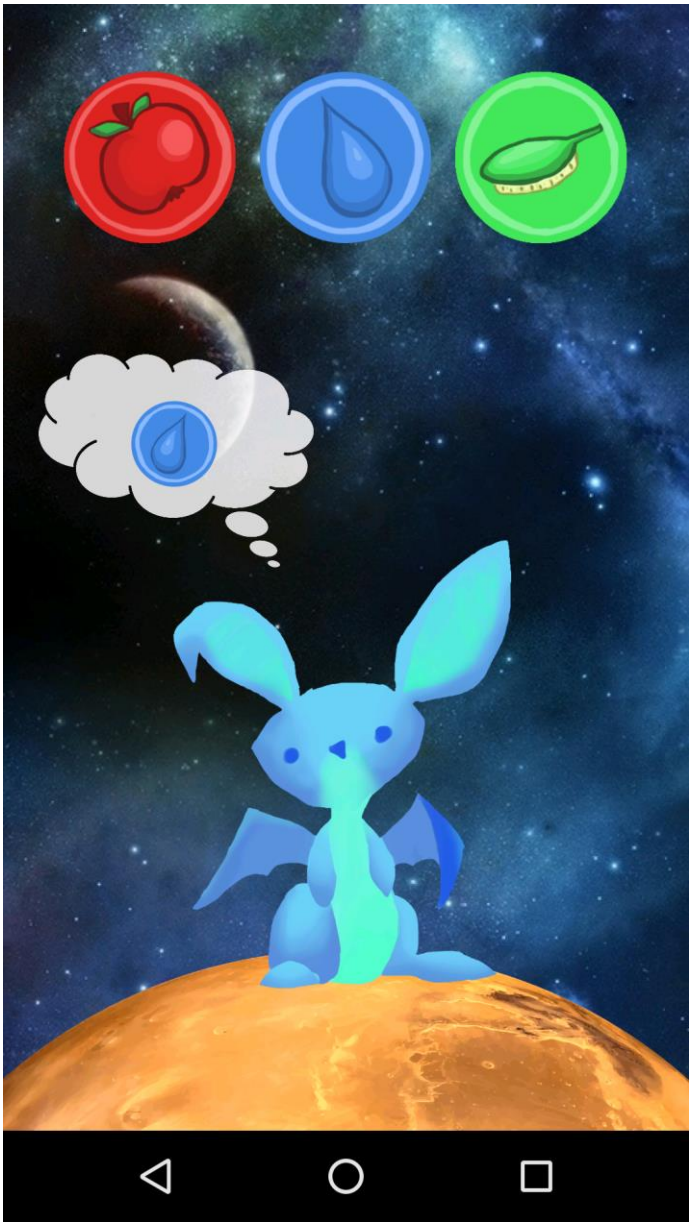
Programmet er et spil som fungerer lidt ligesom en ”Tamagochi”. Dvs. at spilleren har et væsen som de skal tage sig af. Der ses til højre et screenshot af spillet efter det har kørt i noget tid.

Spillet er udviklet som en Android app, dog har jeg implementeret således at det også kan køre på en windows computer.

Problemformulering

Spillet skal som de originale tamagochier, lære børn hvordan de tager sig af væsener (som i de fleste tilfælde vil være kæledyr, eller langt i fremtiden børn). Dette er en aktivitet som børn meget gerne vil efterligne da de ser deres forældre gøre det hele tiden. Spillet skal derfor være nemt at forstå. Spillet må også gerne opfordre spilleren til at lave noget aktivt.

- Hvordan bliver spillet overskueligt og sammentidigt grafisk flot?
- Hvordan kan man integrere en skridtmåler og dermed få spilleren til at være aktiv?
- Hvad er det vigtigste at lære for børn når de skal til at passe deres egne væsener?



Funktionsbeskrivelse

Når spillet starter op vises en velkomst skærm. Den begynder at tælle ned fra 10 sekunder og hvis den når 0 går tutorialen der forklarer hvordan spillet virker automatisk i gang. Der står også at hvis man ønsker at springe tutorialen over skal man ”pet the egg”. Dette betyder at man skal have lært om den game mechanic som er ”pettting/brushing” før man kan springe den over.

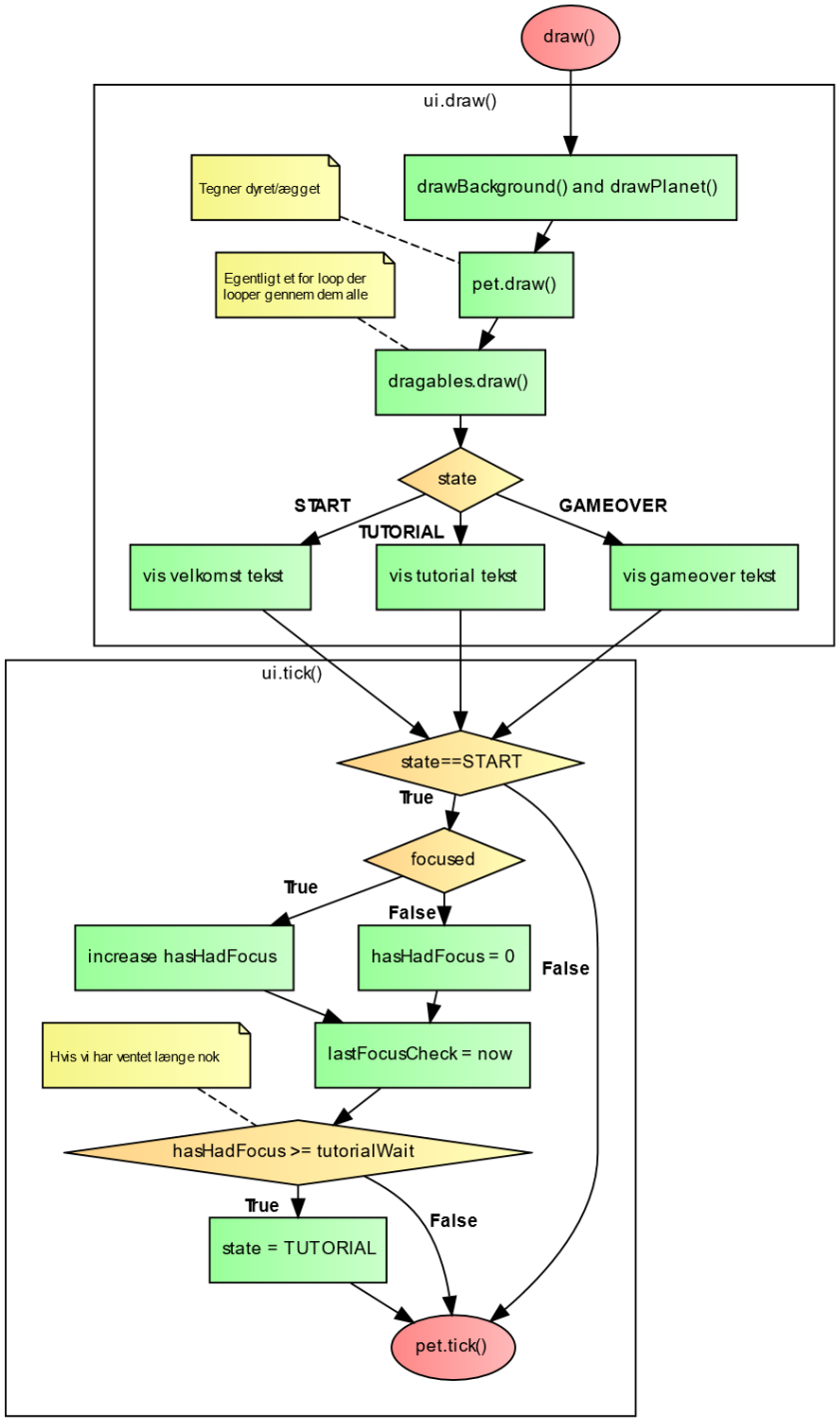
Spillet handler om at man har et væsen (et form for kæledyr eller noget) som lever i rummet. Det starter med at være et æg og har i alt 4 ”stadier”. Hvis man passer det godt vokser det hurtigt op. For at passe væsenet skal man både give det mad, vand, give det opmærksomhed i form af at børste dens pels samt at gå ture med det.

Spillet benytter derfor en skidttæller i din telefon til at se hvor mange skridt du går. For at komme til neste stadie (at ”udvikle”) skal dyret både kunne lide en, og man skal have gået nok. Det er altså ikke nok kun f.eks. at have gået en masse.

Dyret advarer en hvis det er ved at være tørstig eller sulten, og hvis man derefter ignorerer det, vil det langsomt kunne lide en mindre og mindre. Jo mere at dyret/ægget rokker fra side til side jo tættere er det på at udvikle. Dette kan give en god indikation om hvor langt man mangler at gå eller hvor meget man mangler at kunne få det til at lide en.

Teknisk beskrivelse

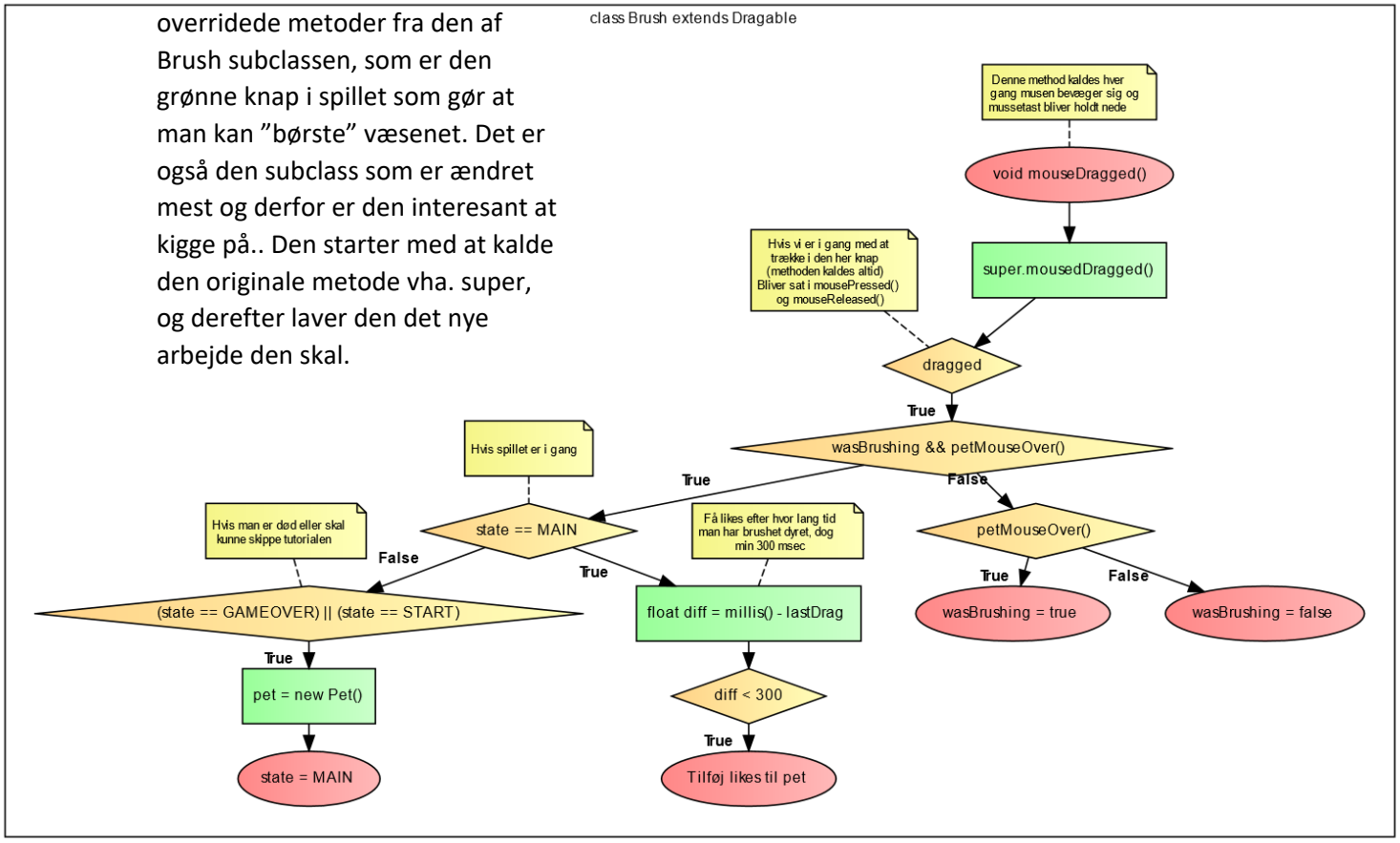
Til højre ses et diagram over et af de vigtigste komponenter i programmet – nemlig den kode der tegner og "ticker" altting. Jeg har valgt såvidt muligt at seperere alt kode der tegner noget (draw) fra alt kode som opdaterer variabler og laver udreninger. Dette gør ikke det store i processing da man kun har en draw() funktion. Hvis jeg havde skrevet det i et andet sprog (eller monkypatchet en masse processing kode) ville jeg kunne låse tick() koden til kun at køre 60 eller 30 gange i sekundet, mens at draw() koden kunne få lov til at køre så meget den nu kan – og derved få bedre framerate. Det gør også koden mere overskueligt.



Det eneste kode jeg har i min draw() er kode der kalder ui.draw og ui.tick – som begge ses til højre.

Draw tegner naturligvis alt hvad der nu skal tegnes. Først baggrund, derefter mellemgrund (dyret) og til sidst GUI elementer. I tick tjekker den først om vi er i START tilstanden. Den tilstand er vi kun i hvis spillet netop lige er starter og tutorialen ikke er blevet sprunget over eller startet. Her kører kode der tester hvor lang tid vinduet har haft fokus – kode der ikke er super relavant på android (men som dog stadig virker) men på PC hvor jeg viser en advarsel når spillet starter (se afsnittet Komplikation vedr. Android) er det vigtigt at den ikke bare fortsætter hvis spillet ligger begravet bag en popup. Hvis vinduet har været fokuseret i lang nok tid (ca. 10 sec) så starter tutorialen (se flowchart på næste side for hvordan koden til at spinge den over fungerer).

Jeg har brugt OOP til at lave min UI. Det har jeg gjort ved at have en abstract base class som hedder Dragable, som alle mine ”Dragables” (altså elementer der kan trækkes – knapperne i toppen) extender. Herunder ses en af de overridede metoder fra den af Brush subclassen, som er den grønne knap i spillet som gør at man kan ”børste” væsenet. Det er også den subclass som er ændret mest og derfor er den interesant at kigge på.. Den starter med at kalde den originale metode vha. super, og derefter laver den det nye arbejde den skal.



Komplikationer vedr. Android

Undervejs i udviklingen af spillet, har der forekommet en del komplikationer af at køre processing på android. De fleste kommer af den render’en på android (Androids egen Android2D) er forskellig fra pc (hvor standarden er JAVA2D men jeg har exporteret spillet med P2D pga. performence problemer ved JAVA2D).

Det første problem jeg stødte ind i, var at jeg gerne ville have haft et såkaldt *spritesheet* til alt mit grafik, og så ville jeg ved opstart ”klippe” den fil i mindre stykker og så scalere dem så de var de rigtige størrelse alt efter hvilken skærmstørrelse spillet bliver afviklet på. Det viste sig dog, at der var en bug som gjorde at resize ikke virker hvis du har lavet dit image via processing. Hvis du loader en billedefil direkte og forsøger virker det helt fint (det var også det jeg endte med at gøre).

Under udviklingen af programmet blev jeg også lidt frustreret over hvor lang tid det tager at compile når man tester på android. Det er meningen at man skal kunne teste det ved hjælp af en emulator, men efter at have brugt flere timer på at få sådan en til at virke, endte jeg med at give op.

Det var yderligere rigtig besværligt at skrive til både android og PC på samme tid (noget som jeg meget gerne vil have, da vejleder og censor skal kunne køre det nemt). Jeg kunne ikke finde nogen god måde at have kode der automatisk kun kører på den ene eller den anden platform (især hvis det inkluderede imports, da processing preprocessoren automatisk flytter dem).

Kodestil

Jeg har valgt ikke at følge min vejleders manifest over kodestil. Det har jeg valgt da baseret på dybdegående research har jeg fundet ud af at langt størstedelen af programmøre mener at det vigtigste når det kommer til kodestil er at være konsistent. Da jeg skriver i processing har jeg valgt at følge deres guide: https://github.com/processing/processing/wiki/Style-Guidelines (jeg er godt klar over at denne guide er til processings interne kode, men mange dele har stadig betydning). Hvis jeg laver en funktion og kalder den ”BezierThing” og processings egne functioner hedder noget i stil med ”bezierCurve” bliver det meget hurtigt forvirrende.

Test

Da koden primært er lavet til at køre på Android har jeg valgt at fokusere mine tests her. Jeg har spillet hele spillet igennem flere gange for at se om koden til at skrifte stadier osv. fungerer som det skal. Jeg har også testet hvor mange skridt der egentlig skal til (altså hvor præcis sensoren er), og det virker som om at det på min egen telefon passer fint nok. Jeg har dog valgt at reducere det totale antal krævne skridt i mine test builds, da vejleder og censor skal kunne spille det uden at skulle vente en masse.

Konklusion

Jeg har i løbet af arbejdet med dette projekt, formådet at lave et spil som fungerer både som app og også som program på en pc. Jeg har gjort spillet grafisk flot ved at bruge mange billeder og flotte animationer. Spillet bruger en skridttæller og opforderer derfor spilleren til at være mere aktiv end de måske ellers ville være.

Spillet lærer børn at det er vigtigt at fodre og give vand til deres kæledyr, og kan måske endda huske dem på at de selv har brug for næring. For at væsenet udvikler sig skal de også tage sig godt af det i form af at børste dets pels. Altså lærer de at dyr har brug for opmærksomhed for at vokse ordenligt op.

Bilag

Herunder her hele programmets kildekode. Bemærk at for at det kan køre i processing skal enten *ANDROID ONLY* **eller** *PC ONLY* kodestykkerne kommenteres ud, alt efter hvilken platform det skal køre på.

Gochi.pde (main)

```
// Gochi by Jacob Bom
// NOTE: this source code contains both Android and PC version.
// Comments are written using we and I pronouns but everything is written by me alone nonetheless
// To switch between them, uncomment the appropiate ANDROID/PC ONLY segments throughout the code.
// Also see the file/tab android_workarounds for Ani workaround.

// This is android only, but it doesn't crash on PC (provided you have
// the correct library) so therefore we just keep it here always
import ketai.sensors.*;

// For prettier animation
import de.looksgood.ani.*;

// For warning if you're not using android
// --- PC ONLY ---
import javax.swing.JDialog;
import javax.swing.JOptionPane;
// --- END PC ONLY

// Only used on android
KetaiSensor sensor;

// What are we doing right now?
// Check ENUMS.java for possible values
State state;
// Our "avatar" so to speak
Pet pet;
// Handles everything that gets drawn
UI ui;
// Handles images
Sprites sprites;

// A "constant" that I use to size things
float wh;

ArrayList<Ani> anis; // see android_workarounds tab for info
ArrayList anisToUnregister; // see android_workarounds tab for info

void setup() {
  println("Starting...");

  // --- PC ONLY ---
  // 540x960 is a pretty good 16:9 mobile size
  // I highly recommend using the P2D render for this program
  // (in fact some things will only work with it)
  // since it relies a lot on images, which the default render is quite horrendous at
  // Default render is faster to launch though, which is why I use it for debugging
  // Fullscreen on a desktop pc is NOT recommended as it does not handle landscape very well

  // NOTE: I moved size() and such to settings() since I needed to do advanced stuff
  // --- END PC ONLY ---

  // --- ANDROID ONLY ---
  // Make it "fullScreen"
  size(displayWidth, displayHeight);
  // Lock it in portrait, otherwise the program will reboot if you tilt the phone
  orientation(PORTRAIT);
  // Init the sensor
  sensor = new KetaiSensor(this);
  // Start collecting data
  sensor.start();
  // --- END ANDROID ONLY ---

  // Don't need any strokes (and if I do in a special case I can just turn it back on)
  noStroke();
  // Should be on by default, but sometimes not on android.
  smooth();
  // Prefer this colorMode for most things
  // makes calls to fill and stroke be (HUE (degrees), saturation (percent), brightness (percent), opacity (percent))
  // Not that many colours are used after I added images, but it's still nice to have.
  colorMode(HSB, 360, 100, 100, 100);
  // I like to use centers for everything
  rectMode(CENTER);
```

```
74    imageMode(CENTER);
75    // I cannot phantom why this is not default
76    ellipseMode(RADIUS);
77
78    // Used to size pretty much all elements
79    wh = (width + height) / 2;
80
81    // We start in this state (see inside UIs draw for what this means)
82    state = State.START;
83
84    // Ani needs a reference to the main PApplet to work, so we supply it here
85    Ani.init(this);
86
87    // Init own classes (make an object of them)
88    sprites = new Sprites();
89    ui = new UI();
90    pet = new Pet(Stage.EGG, false);
91
92    anis = new ArrayList<Ani>(); // see android_workarounds tab for info
93    anisToUnregister = new ArrayList(); // see android_workarounds tab for info
94
95    // Code to check which render is in use and warn about game mechanics if not on android
96    // This is therefore
97    // --- PC ONLY
98    println("Checking sketch render");
99    String renderer = sketchRenderer();
100   println("Using " + renderer);
101   if (renderer != "processing.core.PGraphicsAndroid2D") {
102     String msg = "Dette program er lavet til at køre på Android.\n" +
103                  "Det kan godt køre på din PC, men du får ikke den fulde oplevelse.\n" +
104                  "F.eks. har din computer, modsat android telefoner, ingen skidttæller.\n" +
105                  "Det betyder at du i stedet for at tage skridt bliver nødt til at trykke på 's' knappen\nen
masse gange for at emulere at du tager skridt i den virkelige verden.\n" +
106                  "Tryk på OK for at fjerne denne besked.\n";
107     showPopup(msg, "STOP! VIGTIGT!", JOptionPane.WARNING_MESSAGE);
108   }
109   // --- END PC ONLY ---
110
111   println("Setup complete");
112 }
113
114 void draw() {
115   // Android doesn't have a surface (window)
116   // --- PC ONLY ---
117   surface.setTitle(String.format("Gochi by Jacob Bom [%1$.3ffps] [Use 's' to emulate a step]", frameRate));
118   // --- END PC ONLY ---
119
120   // We draw the bg in UI instead
121   //background(0);
122
123   // First draw then tick (order doesn't really matter since processing doesn't actually draw anything before
PApplet.draw() finishes
124   ui.draw();
125   ui.tick();
126
127   updateAnis(); // see android_workarounds tab for info
128 }
129
130 void keyPressed() {
131   // DEBUG
132   if (key == 's') pet.step(pet.steps+1+pet.initialSteps);
133   else if (key == 'w') pet.waterAdd(1);
134   else if (key == 'f') pet.foodAdd(1);
135   else if (key == 'l') pet.likesAdd(1);
136 }
137
138 // We simply propagate these events to the proper class
139 void mousePressed() {
140   ui.mousePressed();
141 }
142 void mouseReleased() {
143   ui.mouseReleased();
144 }
145 void mouseDragged() {
146   ui.mouseDragged();
147 }
148
149 // This gets called by ketai whenever there is a sensor update.
150 // It is therefore ANDROID ONLY, but it's safe to leave it here even on PC.
151 void onStepCounterEvent(float s) {
152   pet.step(s);
153 }
154
155 // --- PC ONLY ---
156 void settings() {
```

```
  // We have this here due to wanting to have a proper icon
  // And PJOGL is only defined here
  // See comments about size and such in setup()!

  //size(540, 960);
  //fullScreen(P2D);
  size(540, 960, P2D);
  PJOGL.setIcon("icon.png");
}
// --- END PC ONLY
```

Enums.java

```
enum State {
  START, TUTORIAL, MAIN, GAMEOVER
}

enum Stage {
  EGG, BABY, ADULT, DEAD;

  // The last value is so we don't have to implement a ton of edge cases when we're dead
  // Makes animation very slow (though we do that artificially in Pet too) and the rockingStray very low
  // They can be private as they are only used locally (and the java convention is to do that)
  private int[] stepGoals = {10, 30, 50, Integer.MAX_VALUE};
  private int[] likeGoals = {1, 20, 50, Integer.MAX_VALUE};

  // To convert a stage into fx a filename
  // Converts EGG to Egg.
  public String toString() {
      return name().toLowerCase();
  }

  // Gets how many steps it took to get here
  public int prevStepGoal() {
    int goal = 0;
    // ordinal() is the numerical index of the enum (fx. ADULT's ordinal is 2)
    for (int i = 0; i<ordinal(); i++) {
      goal += stepGoals[i];
    }
    return goal;
  }

  // Ditto but for likeability points
  public int prevLikeGoal() {
    int goal = 0;
    for (int i = 0; i<ordinal(); i++) {
      goal += likeGoals[i];
    }
    return goal;
  }

  // Gets how many steps we need to advance to next stage
  public int stepGoal() {
    return stepGoals[ordinal()];
  }

  // Ditto but for likeability points
  public int likeGoal() {
    return likeGoals[ordinal()];
  }
}
```

Pet.pde

```
class Pet {
  // How grown is the pet (see ENUMS.java for different stages)
  Stage stage;
  // Determines the height of the pet
  float size;
  // Base food value (a double due to bad float precission - and we substact a tiny amount each frame - it adds
up)
  double food;
  // Base hydration level
  double water;
  // How many steps the player has taken this stage
  float steps = 0;
  // Where did we start
  float initialSteps = -1;

  // We start this at 1 so that you don't need likes to get out of egg stage
  // (and to avoid dividing by zero and breaking maths)
  float likes = 1;

  // The egg's position
  PVector pos;

```

```
22  // Rocking Animation
23  Ani ani;
24  // Value controlled by Ani to control rocking animation
25  float r = 0;
26  // Is the rocking moving right?
27  boolean movingRight = false;
28  // How much "stray"/movement the rocking animation has
29  int rockingStray = 30+50;
30
31  // Evolve animation index
32  // If it's not -1 then it will start
33  int evolveAniI = -1;
34
35  // Ani for the heart that appears when you gain "likes"
36  Ani heartAni;
37  float heartR;
38
39  // How quickly should food and water go down?
40  double hungerRate = 0.0001;
41  double thirstRate = 0.0005;
42
43  // Moved this instead of inside the tick method so we can debug it easily
44  float progress;
45
46  Pet(Stage stage_, boolean showAnimation) {
47    // Better to delegate to a method
48    changeStage(stage_, showAnimation);
49    // Start in the middle (it's out of 10, though it can technically go
50    // into the negatives, and there you'll start losing likes)
51    food = 5;
52    water = 5;
53
54    // The bottom point (anchor) of the pet
55    pos = new PVector(width/2, height-wh/6);
56    // The height was originally 2 but I upped it since the pictures have paddding (since they aren't the same
    size)
57    size = wh/1.5;
58
59    // Start Rocking animation
60    ani = new Ani(this, 4, "r", 100, Ani.SINE_IN_OUT);
61
62    // Prepare heart animation
63    Ani.noAutostart();
64    heartAni = new Ani(this, 2, "heartR", 100, Ani.QUART_OUT);
65    Ani.autostart();
66  }
67
68  void draw() {
69    pushMatrix();
70    // Translate to bottom point of pet and rotate around that
71    translate(pos.x, pos.y);
72    if (movingRight) {
73      rotate(map(r, 0, 100, -PI/rockingStray, PI/rockingStray));
74    } else {
75      rotate(map(r, 0, 100, PI/rockingStray, -PI/rockingStray));
76    }
77
78    // We + some stuff cause our pictures have slight padding at the bottom
79    float y = -size/2 + size*0.08;
80    image(sprites.pet.get(stage), 0, y); //<>//
81    popMatrix();
82
83    // If we're evolving show the proper frame
84    // We divide by 15 so that we change frame every 15 actual frames
85    // Which makes the animation take ~3/4 of a sec
86    if (evolveAniI != -1) {
87      image(sprites.poof[evolveAniI/15], pos.x, pos.y+y);
88    }
89
90    // Show heart if heart needs to be shown
91    if (heartAni.isPlaying()) {
92      // Make it red, and fade it out
93      fill(0, 80, 80, 100-heartR);
94      // Show it
95      heart(pos.x*1.2, pos.y-size/2-(wh/2/100)*heartR, wh/200, false);
96    }
97
98    pushMatrix();
99    // If we're hungry show a thought bubble with apple in it
100   // That might be better fit to be in UI,
101   // but it has more to do with the pet I'd say
102   if (food <= 2) {
103     image(sprites.thought, width*0.75, height*0.4);
104     image(sprites.smallFood, width*0.75, height*0.39);
105   }
```

```
106     if (water <= 2) {
107       // Flip it
108       scale(-1, 1);
109       // Note the negative x coord due to the scaling
110       image(sprites.thought, -width*0.25, height*0.4);
111       image(sprites.smallWater, -width*0.25, height*0.39);
112     }
113     popMatrix();
114   }
115
116   // So we don't have to always say if we wanna show the animation
117   void changeStage(Stage newStage) {
118     changeStage(newStage, true);
119   }
120
121   void changeStage(Stage newStage, boolean showAnimation) {
122     // If it's dead make it move slowly and change state to GAMEOVER
123     if (newStage == Stage.DEAD) {
124       ani.setDuration(30);
125       state = State.GAMEOVER;
126     }
127
128     // If we should animate set the animation frame to 0 (and therefore not -1)
129     if (showAnimation) evolveAniI = 0;
130     // Actually update the stage
131     stage = newStage;
132   }
133
134   void tick() {
135     // How far are we towards our goal?
136     // Calculated based on both steps and likeability points
137     // We want a number between 0 and 1 even if we are at later stages so we have to factor in how many steps
    and points it took to get here
138     float stepProgress = constrain(((steps-initialSteps)-stage.prevStepGoal()) / stage.stepGoal(), 0, 1);
139     float likeProgress = constrain((likes-stage.prevLikeGoal()) / stage.likeGoal(), 0, 1);
140     progress = (stepProgress + likeProgress) / 2;
141
142     // Evolve once we have enough progress
143     if (progress >= 1.0 && evolveAniI == -1) { //<>//
144       if (stage == Stage.EGG) {
145         changeStage(Stage.BABY);
146       } else if (stage == Stage.BABY) {
147         changeStage(Stage.ADULT);
148       } else if (stage == Stage.ADULT) {
149         changeStage(Stage.DEAD);
150       }
151     }
152
153     // Control rocking animation
154     // We need to do this in the middle so that the egg/pet doesn't just suddently jerk a little towards the
    middle due to higher stray
155     if (ani.getSeek() >= 0.48 && ani.getSeek() <= 0.52) {
156       rockingStray = (int) map(progress, 0, 1, 30+50, 30);
157     }
158     // If the rocking animation reached the end
159     if (ani.isEnded()) {
160       // If we are not dead then change the animation timing so it's more violent the closer to "evolution" you
    are
161       if (stage != Stage.DEAD) {
162         // Do this after ani had ended to avoid jerks due to timing shifts
163         ani.setDuration(map(progress, 0, 1, 4, 0.3));
164       }
165       // Start moving the other way
166       ani.start();
167       movingRight = !movingRight;
168     }
169
170     // If we trigged the evolve animation by setting it to another value than -1
171     if (evolveAniI != -1) {
172       evolveAniI++;
173       // Once we're through reset it back to -1
174       if (evolveAniI/15 >= sprites.poof.length) {
175         evolveAniI = -1;
176       }
177     }
178
179     // If we're starving
180     if (food <= 0) {
181       // Then for each 1 food point that we get below 0 remove a like
182       if ((food % 1) < ((food - hungerRate) % 1)) {
183         likes -= 1;
184       }
185     }
186     // If we're starving of thirst (sidenote: what the hell is that called??)
187     if (water <= 0) {
```

```
188       // Then for each 1 water point that we get below 0 remove half a like
189       if ((water % 1) < ((water - thirstRate) % 1)) {
190         likes -= 0.5;
191       }
192     }
193     if (likes < 1) likes = 1; // We don't wanna accidently divide by zero
194
195     // Actually decrease food and water by their respective rates
196     food -= hungerRate;
197     water -= thirstRate;
198
199
200   }
201
202   boolean collision(PVector otherPos, float otherR) {
203     // The middle of the pet (the different stages have different middles/sizes)
204     // Based on the bottom anchor point (pos)
205     PVector middlePos;
206     float collisionR;
207     if (stage == Stage.EGG) {
208       middlePos = new PVector(0, -size*0.32);
209       collisionR = size/3;
210     } else if (stage == Stage.BABY) {
211       middlePos = new PVector(0, -size/4);
212       collisionR = size/4;
213     } else if (stage == Stage.ADULT) {
214       middlePos = new PVector(0, -size*0.4);
215       collisionR = size/3;
216     } else if (stage == Stage.DEAD) {
217       middlePos = new PVector(0, -size*0.2);
218       collisionR = size/4;
219     } else { //WTF! this should never happen, but java needs it to be happy
220       middlePos = new PVector();
221       collisionR = 0;
222     }
223
224     // Visualize hitbox
225     // Requires repeating invocation of this method
226     //ellipse(PVector.add(pos, middlePos).x, PVector.add(pos, middlePos).y, collisionR + otherR, collisionR +
otherR);
227
228     // Returns a boolean by testing if the dist is greater than the combined radius
229     return PVector.add(pos, middlePos).dist(otherPos) <= collisionR + otherR;
230   }
231
232   void showHeart() {
233     // Rewind the animation and start it
234     heartAni.seek(0);
235     heartAni.start();
236   }
237
238   void step(float steps_) {
239     // If the step sensor was already running on the phone it will report a number of steps
240     // that've been taking without having ever opened the program
241     // So we have to compensate by saving that initial number
242     if (initialSteps == -1) {
243       initialSteps = steps_;
244     } else {
245       steps = steps_;
246     }
247   }
248
249   void foodAdd(float food_) {
250     // Add food but not beyond 10 and also add a like
251     if (food + food_ <= 10) {
252       food += food_;
253       likesAdd(1);
254     }
255   }
256
257   void waterAdd(float water_) {
258     // Add water but not beyond 10 and also add half a like
259     // (thirstRate is a lot faster than hungerRate so its only fair)
260     if (water + water_ <= 10) {
261       water += water_;
262       likesAdd(0.5);
263     }
264   }
265
266   void likesAdd(float likes_) {
267     // If the likes we are adding takes us above the next 0.25 or we're adding 0.25 or more
268     if (((likes % 0.25) > ((likes + likes_) % 0.25)) || likes_ >= 0.25) {
269       // Then show a heart
270       showHeart();
271     }
```

```
    likes += likes_;
  }
}
```

Sprites.pde

```
// Allows us to refrence a sprite just as sprite.poof etc.

class Sprites {
  PImage[] poof = new PImage[3];

  // Maps a picture to each Stage
  HashMap<Stage,PImage> pet = new HashMap<Stage,PImage>();

  PImage bg, planet;
  PImage thought;
  PImage food, water, brush;
  PImage smallFood, smallWater;

  Sprites() {
    // Poof sprites
    for (int i = 0; i<poof.length; i++) {
      // I originally wanted to use a spritesheet for these,
      // but there's a bug in android that means I can't resize an
      // image that processing has generated so I had to do it this way
      poof[i] = loadImage("poof_" + i + ".png");
      poof[i].resize(int(234*(1+wh/600)), 0);
    }

    // Main pet sprites
    for (Stage s : Stage.values()) {
      PImage tmp;
      tmp = loadImage(s.toString() + ".png");
      tmp.resize(int(wh/1.5), 0);
      pet.put(s, tmp);
    }

    // BG
    bg = loadImage("bg3.png");

    // I originally had the BG rotating since turned to be too cpu intensive :(
    //bg.resize((height)*2+width/2, 0);

    // Resize so the picture is the big enough to fit the entire screen (figure out which direction needs scaling)
    if ((width/height) < (bg.width/bg.height)) {
      bg.resize(0, height);
    } else {
      bg.resize(width, 0);
    }
    // Crop to screen size so it works with the background() method
    bg = bg.get(0, 0, width, height);

    // Planet
    // Orignally wanted this to spin too, so I had the whole planet
    // Now I just use a cutout of the top since drawing the rest (even if out of screen) is CPU intensive
    planet = loadImage("planet_top.png");
    planet.resize(int(width*1.5), 0);

    // Thought bubble
    thought = loadImage("thought2.png");
    thought.resize(int(width/2.3), 0);

    // UI Elements
    food = loadImage("food.png");
    food.resize(width/4, 0);
    water = loadImage("water.png");
    water.resize(width/4, 0);
    brush = loadImage("brush.png");
    brush.resize(width/4, 0);

    // Small UI elements for use in thought bubble
    smallFood = loadImage("food.png");
    smallFood.resize(width/8, 0);
    smallWater = loadImage("water.png");
    smallWater.resize(width/8, 0);
  }
}
```

UI.pde

```
class UI {
  // Text shown when game is over
  String gameoverText = "Oh noes... maybe keeping a pet in the empty vacuum of space wasn't such a good idea after all. " +
    "To start over please say goodbye to the pet by petting it one last time.";
```

```
5   // Text shown right when opening game
6   String startText = "Welcome to Gochi by Jacob Bom. The tutorial will commence in \n%1$d seconds.\n To skip the tutorial please pet the egg (weird, I know).";
7   // Text to be shown during tutorial
8   String[] tutorialStepsStrings = {"Hello! Meet your new pet! It's currently an egg, but if you treat it well, it will soon hatch!\nTap the egg to continue...",
9     "To hatch, and later evolve, your pet, you will need to take a walk. You will also need to feed it and give it water!\nTap the egg to continue...",
10    "To feed it simply drag the apple to the pet's mouth. Same applies to water.\nTap the egg to continue...",
11    "To make your pet like you even more you can also pet it using the brush. Unlike the apple and water you have to drag and then rub on the pet while dragging.\nTap the egg to continue...",
12    "Note that you of course do not have to feed the egg and such. That would be silly. Just walk a lot!\nTap the egg to start the game..."};
13
14  // Our ui elements all subclass Dragable
15  Dragable[] dragables;
16
17
18  // When did we last check if we have focus
19  float lastFocusCheck = -1;
20  // How long have we had continues focus for?
21  float hasHadFocus = 0;
22  // How long to wait at the beginning before starting the tutorial
23  float tutorialWait = 10000;
24  // How far are we in the tutorial
25  int tutorialStep = 0;
26
27  UI() {
28    // We simply have 3 UI elements
29    dragables = new Dragable[] {new Food(), new Water(), new Brush()};
30  }
31
32  void draw() {
33    // Always need the BGs
34    drawBackground();
35    drawPlanet();
36
37    // And the pet
38    pet.draw();
39    // And the UI
40    for (int i = 0; i<dragables.length; i++) {
41      dragables[i].draw();
42    }
43
44    // Draw text depending on which state we're in
45    // We should be doing complicated calculations á la
46    // https://forum.processing.org/two/discussion/13105/how-to-make-a-string-of-any-length-fit-within-text-box
47    // here, but I just couldn't be arsed
48    textSize(wh/25);
49    textAlign(CENTER, LEFT);
50    fill(350);
51    if (state == State.START) {
52      // Formats the countdown to be seconds and also adds 0.99 sec so it doesn't ever show 0sec remaining
53      text(String.format(startText, int((tutorialWait - hasHadFocus)/1000+0.99)), width/2, height/10*5, width/10*9, height/10*5);
54    } else if (state == State.TUTORIAL) {
55      text(tutorialStepsStrings[tutorialStep], width/2, height/10*5, width/10*9, height/10*5);
56    } else if (state == State.MAIN) {
57      // We don't have any text here
58    } else if (state == State.GAMEOVER) {
59      text(gameoverText, width/2, height/10*5, width/10*9, height/10*5);
60    }
61
62    // Draw a ton of numbers that are useful when debugging
63    //drawDebugInfo();
64  }
65
66  void tick() {
67    // Makes it so that you have to have continous focus for x sec before the tutorial starts automatically
68    if (state == State.START) {
69      if (focused) {
70        hasHadFocus += millis() - lastFocusCheck;
71      } else {
72        hasHadFocus = 0;
73      }
74      lastFocusCheck = millis();
75      if (hasHadFocus >= tutorialWait) {
76        state = State.TUTORIAL;
77      }
78    }
79
80    // Remember to tick the pet!
81    pet.tick();
82  }
83
```

```
84   void drawBackground() {
85     // This is much faster than image(), but the image has to be the exact dimensions of the canvas
86     background(sprites.bg);
87   }
88
89   void drawPlanet() {
90     // The planet (mars) that the pet is standing on
91     pushMatrix();
92     translate(width/2, height-wh/5+sprites.planet.height/2);
93     // Rotating is too CPU intensive (more into in sprites.pde)
94     //rotate(map(frameCount % (60*80), 0, (60*80), 0, TAU));
95     image(sprites.planet, 0, 0);
96     popMatrix();
97   }
98
99   void drawDebugInfo() {
100    pushMatrix();
101    fill(300, 99, 75);
102    textAlign(LEFT, BOTTOM);
103    textSize(24);
104    text("Steps: " + pet.steps + "\n" +
105      "Likes: " + pet.likes + "\n" +
106      "Progress: " + pet.progress + "\n" +
107      "Food: " + pet.food + "\n" +
108      "Water: " + pet.water + "\n" +
109      "FPS: " + frameRate, 0, height);
110    popMatrix();
111  }
112
113  // Sometimes explicit is better than implicit
114  // Though I will admit there's probably a better way to do these (they are mostly all the same)
115  void mousePressed() {
116    // For each UI element/button
117    for (int i = 0; i<dragables.length; i++) {
118      // Propagate the event
119      dragables[i].mousePressed();
120    }
121  }
122  void mouseReleased() {
123    for (int i = 0; i<dragables.length; i++) {
124      dragables[i].mouseReleased();
125    }
126    // If we're doing the tutorial and mouse is over the pet (and we've released/clicked the mouse)
127    if (state == State.TUTORIAL && (pet.collision(new PVector(mouseX, mouseY), 2))) {
128      // Go to next step in the tutorial (show next string)
129      tutorialStep++;
130      // If we're out of tutorial strings start the actual game
131      if (tutorialStep >= tutorialStepsStrings.length) {
132        state = State.MAIN;
133      }
134    }
135  }
136  void mouseDragged() {
137    for (int i = 0; i<dragables.length; i++) {
138      dragables[i].mouseDragged();
139    }
140  }
141 }
142
143 abstract class Dragable {
144   // Base position
145   PVector startPos;
146   // Current posision (is different from startPos when we're dragging it)
147   PVector pos;
148   // Where on the button did we click?
149   PVector clickOffset;
150   // Radius
151   float size;
152   // Is it currently being dragged?
153   boolean dragged;
154
155   // This is always called by the subclass
156   Dragable(int order, float size_) {
157     // Find a suitable start position so that we can fit 3 elements at the top of the screen
158     startPos = new PVector(width/7*2*order+width/14*3, height/20+width/7);
159     // PVector has a copy() method, but it seems to be missing on android so we use something that works on both
    pc and android
160     pos = new PVector(startPos.x, startPos.y);
161     size = size_;
162   }
163
164   // Overwritten in subclasses
165   abstract void draw();
166
167   // Overwritten in subclasses
```

```
168   // They actually all have the same in them it seems,
169   // but that's because I was thinking that they might have
170   // different shapes and such in the future or something
171   abstract boolean mouseOver();
172
173   // Overwritten in subclasses
174   abstract void callback();
175
176   // Is the mouse over the pet?
177   boolean petMouseOver() {
178     return pet.collision(pos, size);
179   }
180
181   void mousePressed() {
182     // is the mouse over the element/button?
183     if (mouseOver()) {
184       // Well then we're dragging it now
185       dragged = true;
186       // Record where on the button we clicked
187       clickOffset = PVector.sub(startPos, new PVector(mouseX, mouseY));
188       // Fire first drag event ourselves to update position
189       mouseDragged();
190     }
191   }
192   void mouseReleased() {
193     // Were we dragging and is our mouse now over the pet?
194     if (dragged && petMouseOver()) {
195       // Then call the callback (which is implemented in subclass)
196       callback();
197     }
198     // We're no longer dragging
199     dragged = false;
200
201     // PVector has a copy() method, but it seems to be missing on android so we use something that works on both
    pc and android
202     pos = new PVector(startPos.x, startPos.y);
203   }
204   void mouseDragged() {
205     // If we're dragging THIS element then update it's position
206     if (dragged) {
207       pos.x = mouseX + clickOffset.x;
208       pos.y = mouseY + clickOffset.y;
209     }
210   }
211 }
212
213 // Not too much to say about these
214 class Food extends Dragable {
215   Food() {
216     super(0, width/8);
217   }
218
219   void draw() {
220     image(sprites.food, pos.x, pos.y);
221     //pushMatrix();
222     //fill(0, 80, 80);
223     //ellipse(pos.x, pos.y, size, size);
224     //popMatrix();
225   }
226
227   boolean mouseOver() {
228     return dist(mouseX, mouseY, pos.x, pos.y) <= size;
229   }
230
231   void callback() {
232     if (state == State.MAIN) {
233       pet.foodAdd(1);
234     }
235   }
236 }
237
238 class Water extends Dragable {
239   Water() {
240     super(1, width/8);
241   }
242
243   void draw() {
244     image(sprites.water, pos.x, pos.y);
245     //pushMatrix();
246     //fill(215, 80, 80);
247     //ellipse(pos.x, pos.y, size, size);
248     //popMatrix();
249   }
250
251   boolean mouseOver() {
```

```
    return dist(mouseX, mouseY, pos.x, pos.y) <= size;
  }

  void callback() {
    if (state == State.MAIN) {
      pet.waterAdd(1);
    }
  }
}

// This one is more interesting
class Brush extends Dragable {
  // When did we last drag
  float lastDrag;
  // Were we brushing on last drag event?
  boolean wasBrushing;

  Brush() {
    super(2, width/8);
  }

  void draw() {
    image(sprites.brush, pos.x, pos.y);
    //pushMatrix();
    //fill(100, 80, 80);
    //ellipse(pos.x, pos.y, size, size);
    //popMatrix();
  }

  boolean mouseOver() {
    return dist(mouseX, mouseY, pos.x, pos.y) <= size;
  }

  void callback() {
    // Do nothing
  }

  void mouseDragged() {
    // We want the old functionality we just wanna expand upon it slightly
    super.mouseDragged();
    // Make sure that we're dragging THIS element
    if (dragged) {
      // Are we brushing/petting the pet?
      if (wasBrushing && petMouseOver()) {
        if (state == State.MAIN) {
          // Get likability points based on how long you brush
          // I don't want you just to be able to idlily hover the brush over pet
          // So if we don't get a mouseDragged event in 300ms we don't count it
          float diff = millis() - lastDrag;
          if (diff < 300) {
            pet.likesAdd(map(diff, 0, 300, 0, 0.05));
          }
          lastDrag = millis();
        } else if ((state == State.GAMEOVER) || (state == State.START)) {
          // "Brushing the pet" is how you skip the tutorial
          // and how to restart the game after you get gameover
          pet = new Pet(Stage.EGG, true);
          state = State.MAIN;
        }
      } else {
        // If we're over the pet now then we wanna know that next time this get's called
        // Since that means we ARE brushing it
        wasBrushing = petMouseOver();
      }
    }
  }
}
```

utils.pde

```
// Makes a non-modal JOptionPane messagebox
void showPopup(Object msg, String title, int messageType) {
  // --- PC ONLY ---
  // Make the pane
  JOptionPane pane = new JOptionPane(msg, messageType);
  // Make the pane create the dialog for us
  JDialog dialog = pane.createDialog(null, title);
  // Tweak it to be non-modal
  // The reason we want this, is that if it was modal it would halt processings draw thread,
  // which means you see nothing but a white screen while the popup is open
  dialog.setModal(false);
  // Display it
  dialog.show();
  // Put it on top of processings own window
  dialog.setVisible(true);
```

```
  // --- END PC ONLY
}

// Copied from previous project
// YAY code reuse
void heart(float x, float y, float size, boolean half) {
  pushMatrix();
  translate(x-50*size, y-20*size);
  beginShape();
  if (!half) {
    vertex(50*size, 15*size);
    bezierVertex(50*size, -5*size, 90*size, 5*size, 50*size, 40*size); // Right
  }
  vertex(50*size, 15*size);
  bezierVertex(50*size, -5*size, 10*size, 5*size, 50*size, 40*size); // Left
  endShape();
  popMatrix();
}
```

android_workaround.pde

```
// The libraray Ani that I'm using is broken on android due to the lack of a proper registerPre
// The code below (and some bits in the main file) simulate the proper behaviour.

// workaround -----------------------------------------
void updateAnis() {
  if (anis.size() == 0) return;

  for (int i=0; i < anis.size(); i++) {
    Ani aniTmp = (Ani)anis.get(i);
    aniTmp.pre();
  }

  if (anisToUnregister.size() > 0) {
    for (int i=0; i < anisToUnregister.size(); i++) {
      anis.remove(i);
      anisToUnregister.remove(i);
      println("removed");
    }
  }
  println(anis.size());
}

void registerPre(Object obj) {
  anis.add( (Ani)obj );
}

void unregisterPre(Object obj) {
  int index = anis.indexOf(  (Ani)obj );
  anisToUnregister.add(index);
}
```