List of potential refactors

Playing sounds

Author: Luke

An issue I've encountered in several classes has to do with how game sounds are called and played. The sounds list is directly accessed from game and checks for the existence of the sound has to be made in the class using the sound.
The responsibility for looking up and playing the sound should go to the holder of the sounds itself, the Game class. A method in the game class should be made to access and look for the sound to play. This will make testing also easier, as playing sounds will not contribute to the branch counter. An example of the refactor can be found in the SoundPlayRefactor branch.

```
/**
 * die: lets the player die.
 */
private void die() {
      if (game.getSounds().size() > 0) {
            game.getSounds().get(4).play();
      }
      alive = false;
}
```

Example in Player class.

The mess that is the Level class

Author: Luke

This class raises many questions for me. This class gets instanced by game for every level that gets played, yet almost all properties are statically accessed. This indicates that actually only one Level needs to instanced and the contents should be changed within, yet this is not the case.
We need to decide how this class could be implemented most optimally keeping in mind how it's functioning in relation to other classes. Should only one level be instanced and let the level load each new level or should the game class load each unique level by instancing it, thus leaving most of the level attributes unchanged? Changing this class however will require some work.

Duplicate Code in menu classes

Author: Jasper

The classes MainMenu and OptionMenu have quite a lot of duplicate code, for example rendering of the background and buttons. An abstract Menu class should be used to let MainMenu and OptionMenu inherit basic functionality from.

```
public final void render() {
          background.bind();

          GL11.glBegin(GL11.GL_QUADS);
          glColor4f(1.0f, 1.0f, 1.0f, 1.0f);

          GL11.glTexCoord2f(0, 0);
          GL11.glVertex2f(-(float) width / 2, height);

          GL11.glTexCoord2f(1, 0);
          GL11.glVertex2f((float) width / 2, height);

          GL11.glTexCoord2f(1, 1);
          GL11.glVertex2f((float) width / 2, 0);

          GL11.glTexCoord2f(0, 1);
          GL11.glVertex2f(-(float) width / 2, 0);

          GL11.glEnd();

          ....
```

Text rendering

Author: Jasper

The code used to render a single line of text is very long and maybe a Label class could be created to reduce the code needed to render text.

```
GL11.glScalef(1, -1, 1);
      TextureImpl.bindNone();
      font.drawString(
            -font.getWidth("Sound Effect Volume:"),
            (float) -Launcher.getCamHeight() / 3f - 180
            - font.getHeight("Sound Effect  Volume:") / 2f,
            "Sound Effect Volume:", Color.white);
GL11.glScalef(1, -1, 1);
```

```
//Label(Point pos, Font font, String text, boolean CenterWidth,
boolean CenterHeight)

Label lbl = new Label(Point pos, Font font, "Sound Effects
Volume:", true, true);
lbl.render();
```

Example in OptionMenu for test rendering and what it could be

Ball Constructor (and scorepopup)

Author: Bart

The fact that raised attention for these specific class was that we initially could not get rid of an checkstyle error that occurred. This error had to do with magic numbers. After further investigation we might consider instantiating various ball elements with corresponding characteristics.

```
public Ball(final Point pos, final float radius) {
      super(pos, radius);

      switch ((int) radius) {
      case 50:
            setColor(Color.red);
            break;
      case (50 / 2):
            setColor(Color.blue);
            break;
      case (50 / 4):
            setColor(Color.green);
            break;
      default:
            setColor(Color.green);
            break;
      }
      height = getPosY();
    }
```

```
public Ball(final Point pos, final float radius) {
      super(pos, radius);

      switch ((int) radius) {
      case Blue ball:
            {/**
              **Blue ball
              **etc
              **/
   }
```

Example in ball constructor and what it could be

In general: redundant if conditions

Author: Bart

In some classes in the packages objects and shapes there are some if and if..else statements in which the final option contains a condition. We have to look at this and check this for possible redundancies with the authors of the specific classes.

ADT use in Box class

Author: Bart

For example the corners, an array of Point is used. It might be considerable to use an ArrayList with the use of getters and setters to benefit from the functionalities these provide. Encapsulation of behavior associated with getting or setting the property - this allows additional functionality (like validation) to be added more easily later.

HashCode() method in Box

Author: Bart

This is not used. Although this must be overwritten, this could be handled more elegantly. Examples are in slides from SEM on design patterns.

If-else if-else if… in Level.readLevel()

Author: Arjan

There is a long, long, if-else list in this method. A switch would be much more appropriate.

```
if (type.equals("gravity")) {
//code
} else if (type.equals("box")) {
//code
} else if (type.equals("ball")) {
//code
} else if (type.equals("player")) {
//code
} else if (type.equals("name")) {
//code
}
```

Javadoc in multiple classes

Author: Arjan

The javadoc describing the classes collision, collisiondetection and gamesettings does not describe anything concrete at all. Should be more specific on the purpose of the class.

```
/**
 * Class for collision.
 *
 */
public class Collision {
```

If-else if-else if… in collisionDetection.collision() and collideBoxBall()

Author: Arjan

There is a long, long, if-else list in this method. A switch would be much more appropriate.

```
if (side != 0) {
      if (side == 1) {
            side = 3;
      } else if (side == 2) {
            side = 4;
      } else if (side == 3) {
            side = 1;
      } else if (side == 4) {
            side = 2;
      }
```