Drum Machine

We

Can

Do

It

Yes

We

can

Who Will Record?

You press the RECORD button

The metronome ticks

The beat counter counts through the beats of the pattern 0 to 63 over and over

You tap on a drum

Who records it?

We have a variable to hold the sounds in the pattern

List<String> drumTrack = List(LOOP_SIZE);

Somehow the drum that got tapped needs to get put in that list in the right beat location

The only object that knows it's been tapped is the drum itself.

The game controller doesn't keep track of that

The game itself pays no attention to who is getting tapped

We will have the drum itself record the fact that it's been tapped.

Drum will call a method on the drum controller whenever it is tapped

The controller can decide what to do with that information.

Add this method to the drum controller.

```
// called by a drum when it is tapped
void drumTap(Drum drum) {
  if (state == State.RECORD) {
    // add this sound to the loop track
    int beat = currentBeat - 1;
    if (beat < 0) beat = LOOP_SIZE - 1;
    drumTrack[beat] = drum.sound;
  }
}
```

Why do we add 1 to the beat first?

The last thing beat() does after playing the metronome is to increment the beat number.

The beat number is actually the beat we will be on when the beat time is reached

We are recording the beat before. Anytime between the first metronome click and the next beat we want to record at beat 0

We check if we are past the last beat of the pattern and start over if we are

We store the drum sound file name in the list entry for that beat

We only do this if we are in the RECORD state

The drum doesn't care.

It doesn't need to know anything about the working of the controller

Drum

The drum needs to be able to call the drumTap() method on the drum controller.

We need to pass the drum controller object to the drum when creating it so that the drum will have access to that method.

Add a new instance variable and change the Drum constructor

```
final DrumController controller;


Drum(this.game, this.controller, double x, double y, double size, String image,
this.sound) : super.square(size, image) {
```

Add this where the drum realizes it has been tapped.

```
game.wasTapped = false;  // reset tap
Flame.audio.play(sound);
controller.drumTap(this);
```

Finally, everywhere we create a drum object in the drum controller, we have to add the controller as one of the constructor parameters.

```
Drum(game, this, pctX*10, pctY*10, pctX*20, 'drum/frame drum.jpg', 'drum/frame drum.wav'),
```

Do that to all the drum constructors

When should we erase the pattern to start a new one?

When the RECORD button is tapped.

Add this to the code when we handle the RECORD tap

```
// see if user tapped on record button
if (game.wasTapped && button1Rect.contains(Offset(game.tapX,game.tapY))) {
  game.wasTapped = false;  // reset tap
  Flame.audio.play(BUTTON_PRESS_SOUND);
  state = State.RECORD;
  currentBeat = 0;  // start of pattern
  timeNextBeat = game.currentTime() + 1;  // start first beat soon
  metroBeat = 0;  // start ticking on first beat of loop
  drumTrack = List(LOOP_SIZE);  // clear out loop track
}
```

Playing the Pattern

We play the drum pattern in the PLAY and RECORD states

- In the RECORD state so we can hear what we have made so far
- In the PLAY state because we want to hear it without the metronome

The beat() method is where we play the sound that was recorded. Add this after we play the metronome beat to play the sounds while we are recording

```
// play any sound recorded
if (drumTrack[currentBeat] != null) {
  Flame.audio.play(drumTrack[currentBeat]);
}
```

See if you can add the code to play the pattern while we are in the PLAY state

Add this as a new state test in the beat() method.

```
} else if (state == State.PLAY) {
  // play any sound recorded
  if (drumTrack[currentBeat] != null) {
    Flame.audio.play(drumTrack[currentBeat]);
  }
} // state
```

That's it. Play the game – preferably on a real phone

Hit RECORD

As the pattern loops around and around, add different drum sounds to fill it out

When done, press STOP

Press PLAY to play back your pattern.

The code is here:

https://github.com/shawnlg/flutter_ball/tree/12_drum_machine

Just the Beginning

We have a rudimentary drum machine

We could make improvements – whole other app

- Save patterns
- Edit already recorded patterns
- Change the beat speed, number of beats, etc.
- Change time signature – beats per measure, etc.

We are done with drums.

Next time, back to our ball game where we will bounce off of things