Raspberry Pi OpenGLES

the object_container lib, and how operation ordering may affect performance

ribamar santarosa
INdT

Outline

- Part 1 - object_container library

- Introduction
- Architecture
- General Use Case
- Details

- Part 2 - Ops ordering affecting performance

- Optimization points
- The choice of glGenTextures()
- Methodology for analysing performance
- Architecture diff
- Results
- Reproducing tests

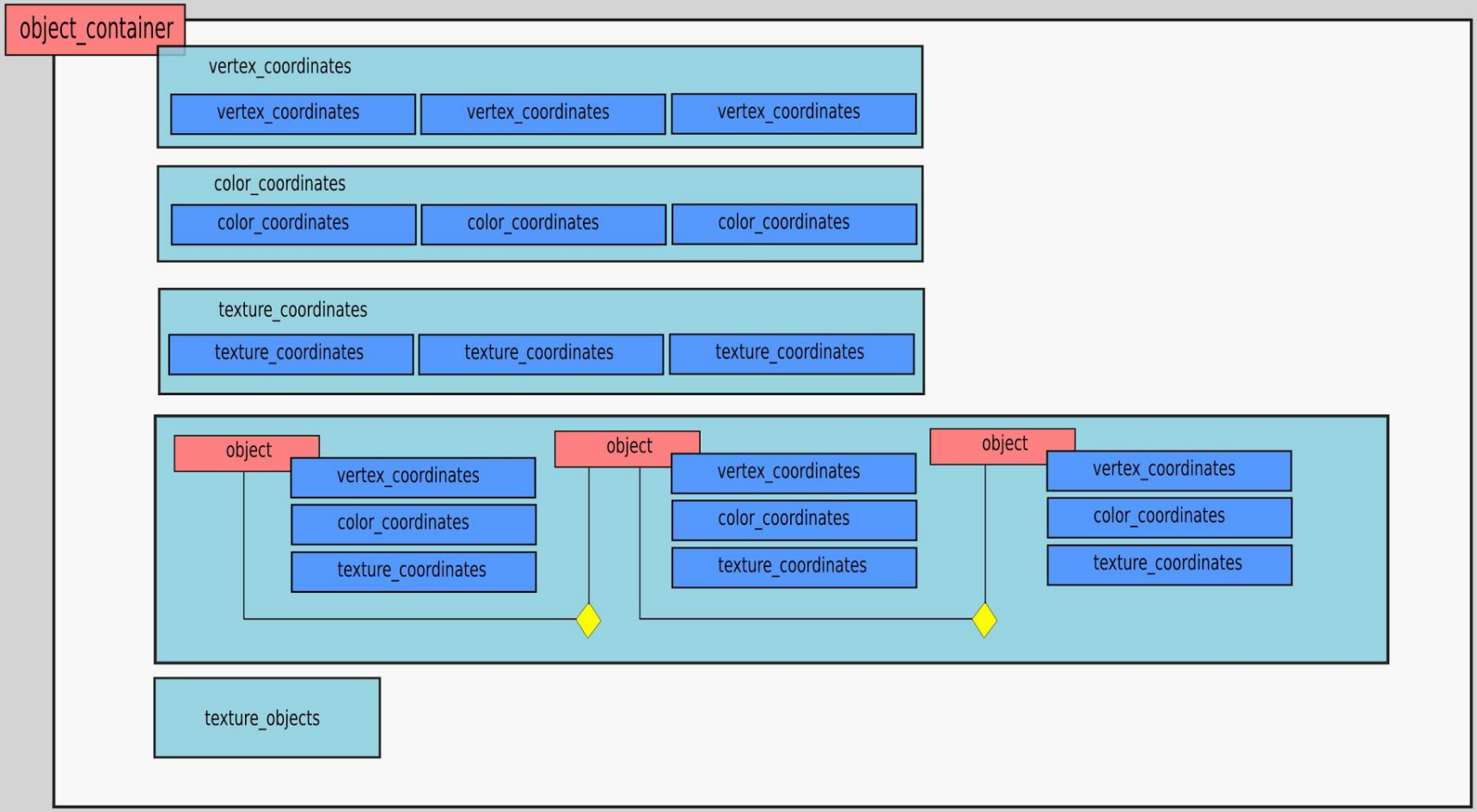
Part 1 - the object_container lib:
[1/4] Introduction

- History: APM 2012 (fonts).

- Motivation: dealing with the complexity of assembling arrays.

- Results: initial idea wasn't completely achieved, but the final output was arguably better than it.

Part 1 - the object_container lib:
[2/4] Architecture



Part 1 - the object_container lib:
[3/4] General Use Case

0. ogles_init(state); /* provided in examples */
1. o = object_new(/* several params */);
2. o_c = object_container_new();
3. object_container_add_object(o_c, o);
4. object_container_prepare_to_draw(o_c);
5. /* intentionally left empty */
6. object_container_draw_objects(o_c);
7. eglSwapBuffers(state->display, state->surface);

Part 1 - the object_container lib: [4/4] In Details

- object_container_prepare_to_draw(): eventual optimizations.

- intentional empty step 5 in the previous slide:
  - object_container_set_translation(); /* untested! */
  - object_container_set_rotation_angle(); /* untested! */
  - user transformations;
  - user optimizations? (maybe too late...)

Part 1 - the object_container lib:
[4/4] In Details

- after object_container_draw_objects() data still on GPU: others transformations are possible, w/o a new object_container_prepare_to_draw().

- parallel object_container's are possible. however, after object_container_prepare_to_draw(o_c), GPU is committed to that o_c -- no object_container_draw_objects(other_o_c) call now.

- still possible to draw multiple o_c to the same frame (both parallel or serialized), while eglSwapBuffers() is not called.

Part 2 - Ops ordering affecting performance:
[1/6] Identified Optimization Points

- Shader related -- example of default shader's stupidity: glRotate(0,0,0,0) is really computed!

- Texture-related:
  - batch texture_objects generation w/ glGenTextures().
  - reuse texture_objects.

- Coordinate-related:
  - pack textured objects before in the texture vertices array (so don't waste memory and band w/ non-textured objects).
  - per-vertex (and not per-coordinate type) coordinate array alignment.

Part 2 - Ops ordering affecting performance:
[1/6] Identified Optimization Points

- State machine related:
  - avoid glEnable()/glDisable() calls

(thanks dakerfp)

Part 2 - Ops ordering affecting performance:
[2/6] The Choice of glGenTextures()

- simple enough.

- batching operations: others claim to optimize canvas by batching gl function calls.

- function signature itself:

```
void glGenTextures(GLsizei n,  GLuint * textures);
```

that **n** isn't there for nothing!

Part 2 - Ops ordering affecting performance:
[3/6] Methodology for analysing perf

- 2 versions of object_container:
  - one that calls glGenTextures() every time a new objects is created (object_new()) -- called "non-batched" version;
  - one that gets all objects added to the object_container, and makes only one call to glGenTextures(), in object_container_prepare_to_draw() -- called "batched version";

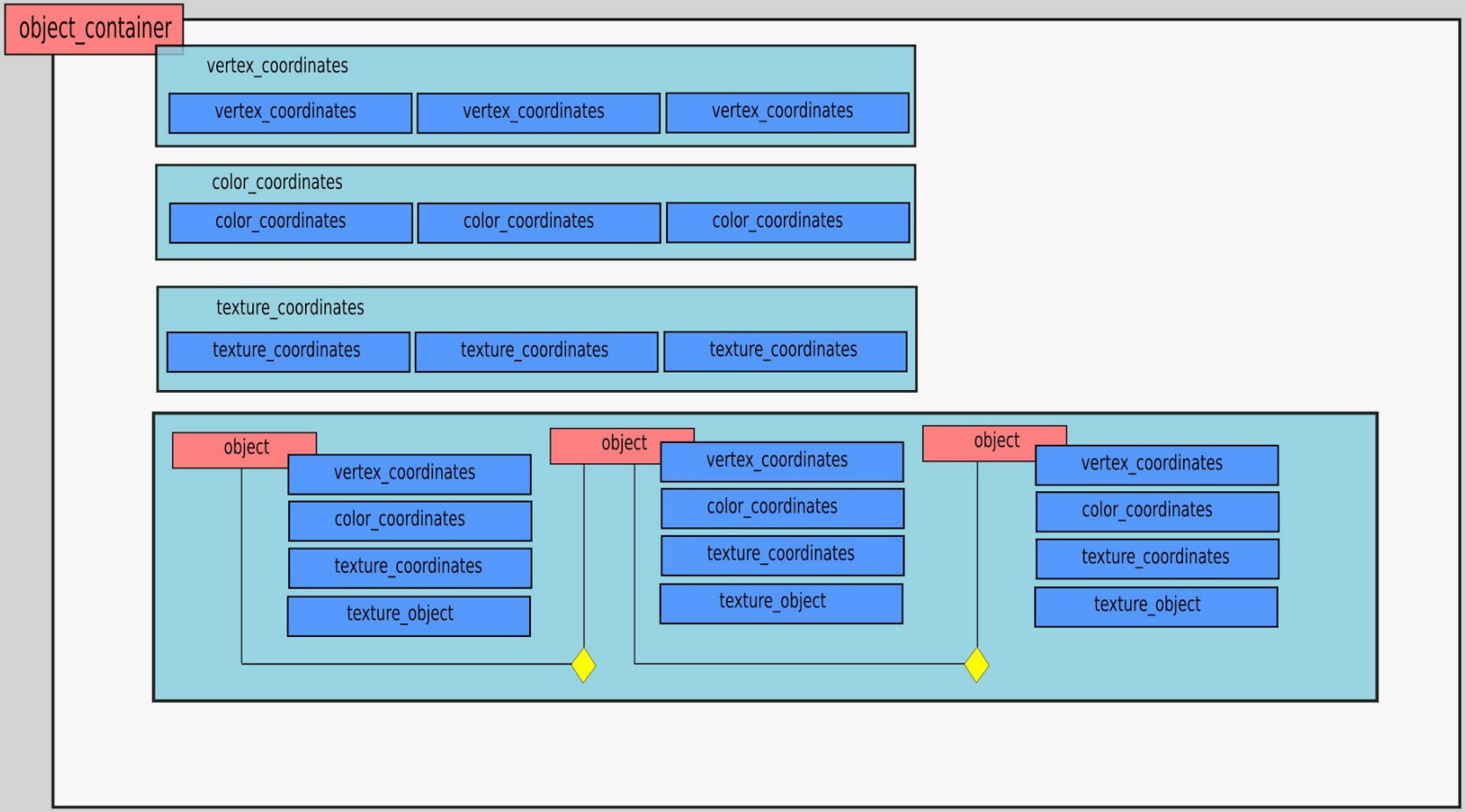
- A sample application, that should be compiled unaltered with both versions of object_container, where:
  - N objects (the simplest ones, so triangles) are created, added to o_c and drawn per frame;

Part 2 - Ops ordering affecting performance:
[3/6] Methodology for analysing perf

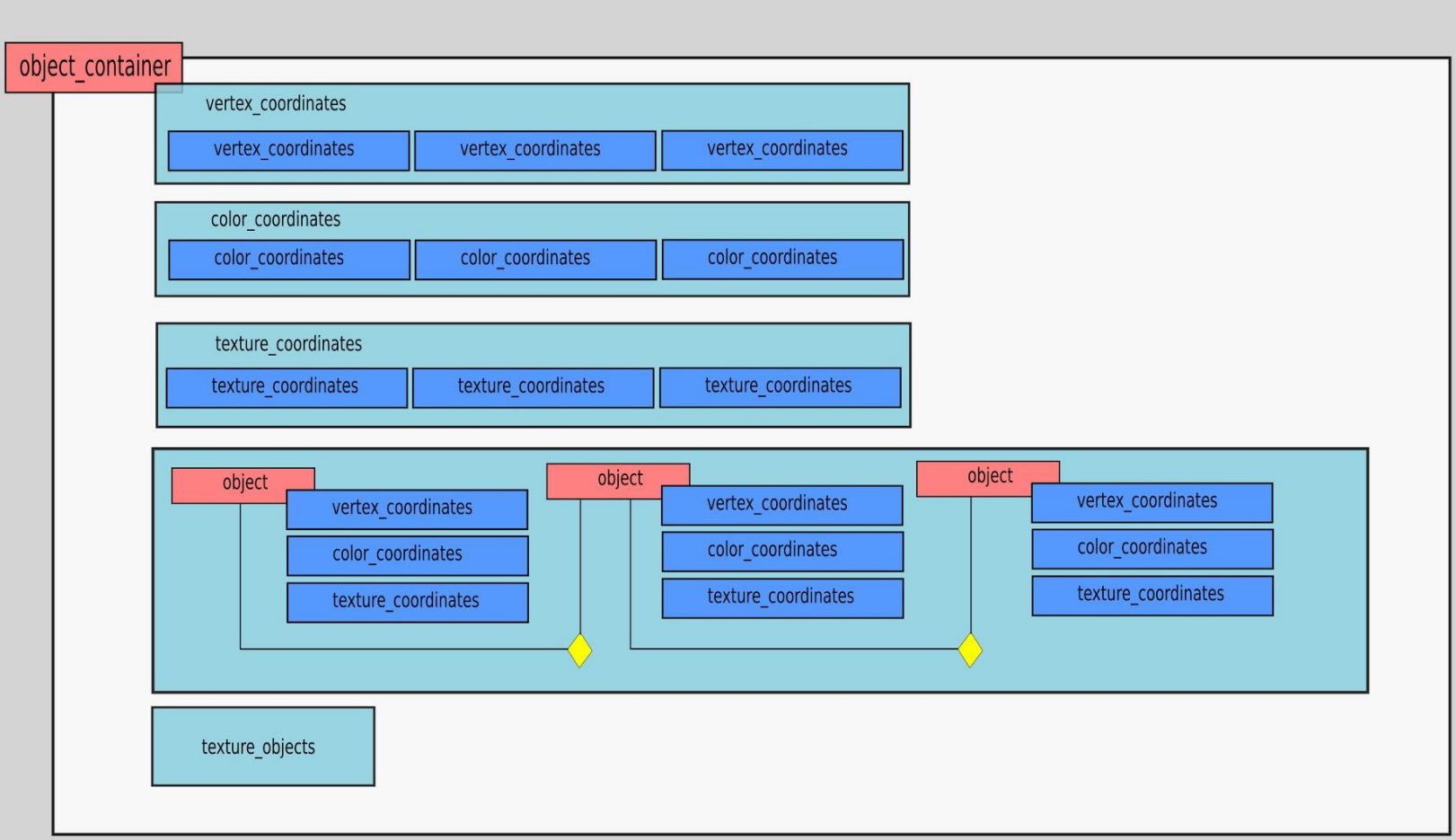
- N big enough to produce sensible diffs, but not too big to overflow processing;
- Their textures should be the same and as simple as possible;
- Idea behind it: perf differences between two executions piped into glGenTextures() (avoid everything else that may be affected by the growing of N, like a complicated texture rendering).

- Perf metrics:
  - frame rate (estimated by the application itself);
  - glGenTextures() (and glDeleteTextures()) timeshare (calculated by perf_3.2 tool).

Part 2 - Ops ordering affecting performance:
[4/6] Architecture Diff -- non-batched



Part 2 - Ops ordering affecting performance:
[4/6] Architecture Diff -- batched



Part 2 - Ops ordering affecting performance: [5/6] Results

Method	# of Triangles	fps	glGenTextures() timeshare	glDeleteTextures() timeshare
Non-batched	10	27/28	0.10%	0.05%
Batched	10	30/31	0.01%	0.00%
Non-batched	50	5/6	0.17%	0.04%
Batched	50	6/7	0.01%	0.01%

Part 2 - Ops ordering affecting performance: [6/6] Reproducing tests

- blog post in ... ?

- hash 495fc6e899e3f264c0df69fe5f6c07fbcd30d608 - non-batched version.

- hash 20ecc77d20a5d1657d887fe3ad0c888e7a707e88 - batched version.

Part 2 - Ops ordering affecting performance: [6/6] Reproducing tests

```
# open to ssh terminals to rpi

make clean ; make head
./example_object_container.bin &> ~/glGenTextures-outputs/batched-out-N10
# control-z

# in the second terminal:
pid=$(ps aux | grep example_object_container.bin | grep -v grep | cut -f 7 -d " ")
perf_3.2 record  -p $pid -g sleep 100 ; kill -9 ${pid}

# immediately back in the first terminal
fg # to bring example_object_container.bin back to the foreground

# results
perf_3.2  report
grep frames ~/glGenTextures-outputs/batched-out-N10
```

It's the end, my frrrriend!

- get it at: github.com/ribamar-santarosa/opengles-object_container

- contact author:

ribamar@openbossa.org or ribamar@gmail.com

questions (find us at yahoo answers)? suggestions (find us at diy.org)? jokes? ...

...corrections?

Tá na hora de comer...

Tá na hora de comer...

... pão de queijo!