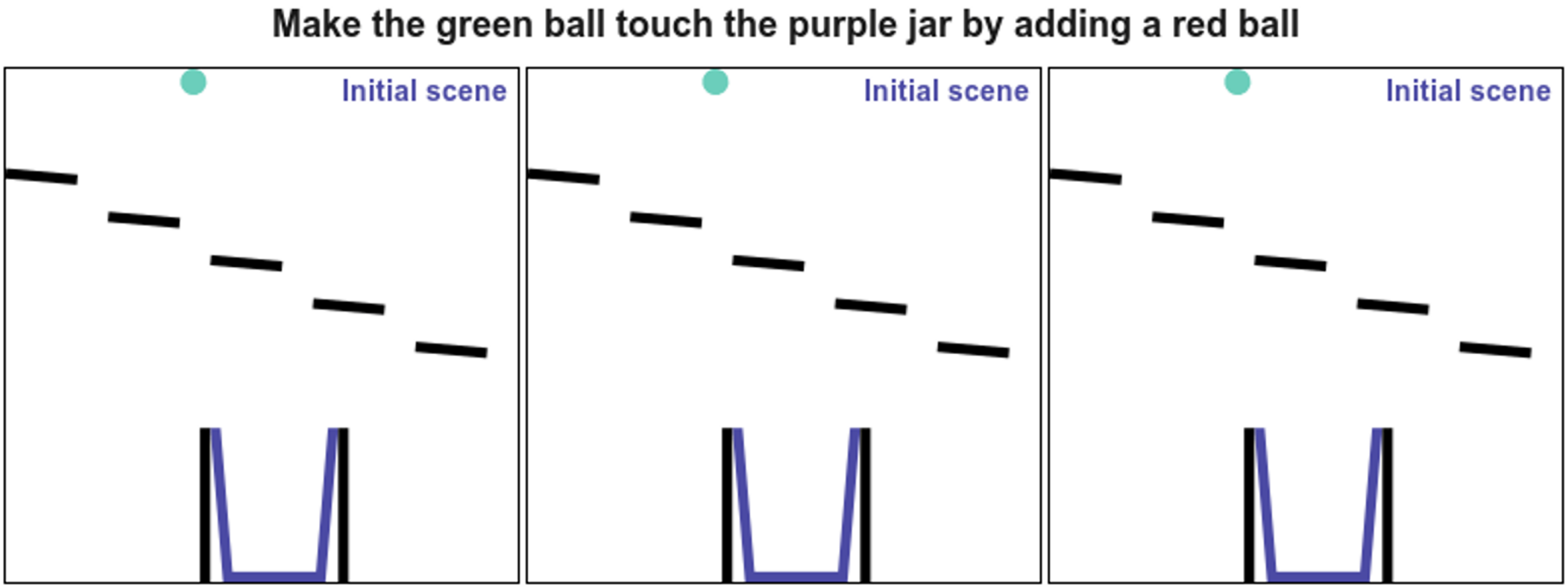
ESPRIT: Explaining Solutions to Physical ReasonIng Tasks

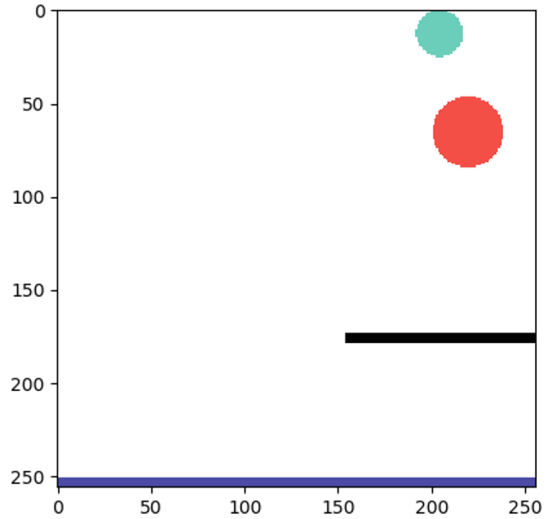
Jeremy Weiss

PHYRE


Make the green ball touch the purple jar by adding a red ball
Initial scene
Initial scene
Initial scene

Approach

- Brute force solution (or non-solution)
- Identify salient events
  - Collisions and separations
  - Large → small



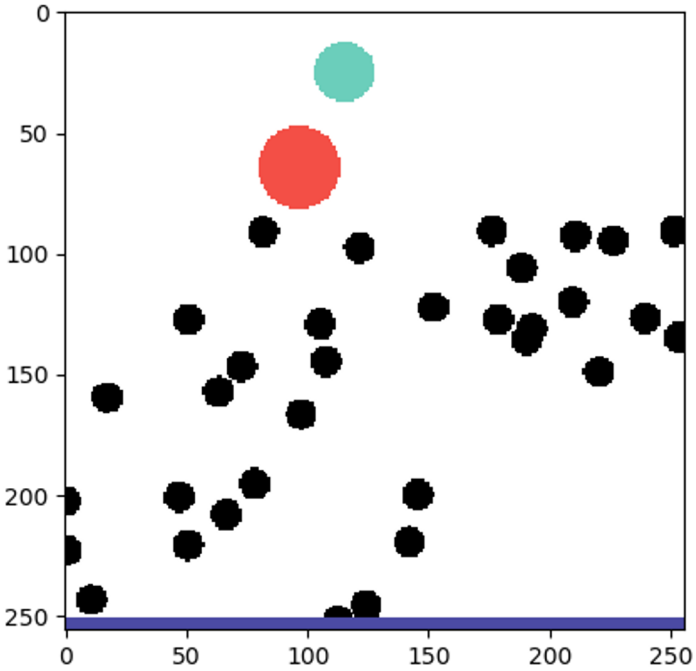
- Salient data (MTurk) + data-to-text = sentences

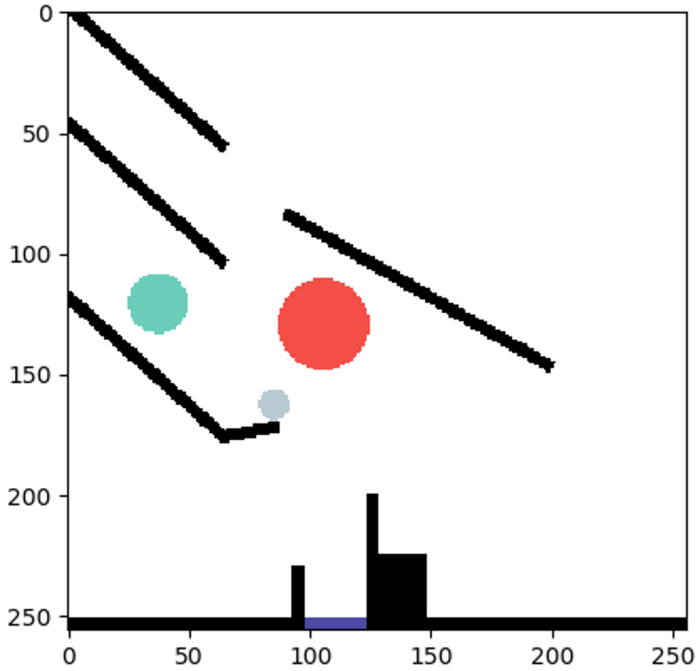
```
class contactAppend : public b2ContactListener

void BeginContact(b2Contact* contact)

void EndContact(b2Contact* contact)
```

Images


0
50
100
150
200
250
0
50
100
150
200
250


0
50
100
150
200
250
0
50
100
150
200
250

```
py::capsule freeCollisions(collisions, [](void *f) {
    auto *foo = reinterpret_cast<uint32_t *>(f);
    delete[] foo;
});
  ;;;

auto collisionsArray =
      py::array_t<uint32_t>({1000}, {sizeof(uint32_t)}, collisions, freeCollisions);

...

PYBIND11_MODULE(simulator_bindings, m)
```

```
status, images             = simulator.simulate_single(task_index, action, stride=1, need_images=True)

status, images, collisions = simulator.simulate_single(task_index, action, stride=1, need_images=True)
```

```
for im, step in ims_needed:
    plt.imshow(im)
    plt.savefig(os.path.join(save_dir, str(step)))
    plt.cla()
```

```python
# Raw data extracted from the simulation
for file in os.listdir("outputs"):
        template = read_json_file("outputs/" + file)

        # Not every task has a solution when bruteforced from a discrete action space
        for task in template:
            if "unsolved" in task:
                continue

            task_id = task["task_id"]
            action = task["action"]
            action, is_valid = simulator.get_user_input(action)
            task_index = tasks.index(task_id)

            # Without searching for the solution, the simulation is quite fast
            result = sim.simulate_task_with_input(simulator.get_task(task_index), action, stride=1)

            # Extract intrinsic properties of objects (radius, length, etc.)
            viz_list = viz.create_list_of_objects(result.sceneList[0])  # Thanks Aadit :)

            solved_states = result.solvedStateList

            user_id = max(task["initial"], key=lambda x: x["id"])["id"] + 1

            # Window filter
            collisions = get_collisions(None, task, 5)  # Thanks Aadit :)

            ... <Data reformatting> ...

            template_num, task_num = task_id.split(":")
            with open(os.path.join("filtered", template_num, f"{task_num}_list.json"), "w") as f:
                json.dump(collisions, f)
```

More data formatting → structured table

step	is_list	id	type	color	x	y	x_vel	...	is_collision	kind	id_1	type_1	...
40	False							...	True	begin	5	bar	...
40	True	9	User Circle	Red	4.8	7.1	0	...	False				
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
378	True	9	User Circle	Red	3.3	5.9	0	...	False				...
378	True	4	circle	Green	21.4	1.9	3.3	...	False				...

Future Work

- Expand data set (only contains first 25 templates)
- Expand action space (multiple/new objects)
- Move away from Box2D simulation
  - Possibly HOG + classifier
- Break assumptions (only trained with solutions)
  - Generalize physical reasoning