Whisker: Automated Testing of Scratch Programs

Marvin Kreis

Chair of Software Engineering II
University of Passau

2019-03-27

What is Scratch?

What is Scratch?

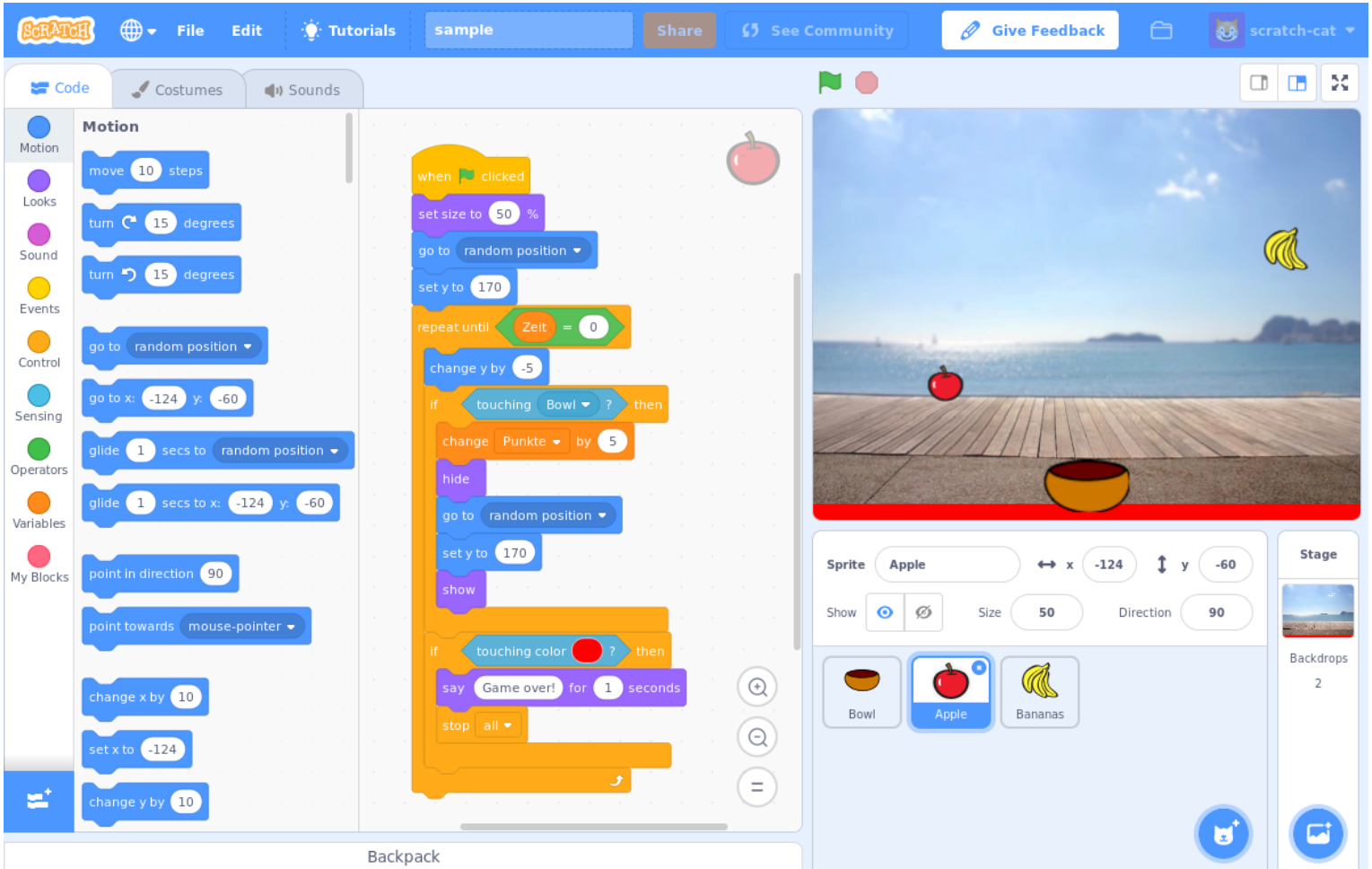


Figure: Scratch's GUI

What is Scratch?

- Block-based programming language
- Developed by the MIT media lab
- Code is separated into scripts that are triggered by events

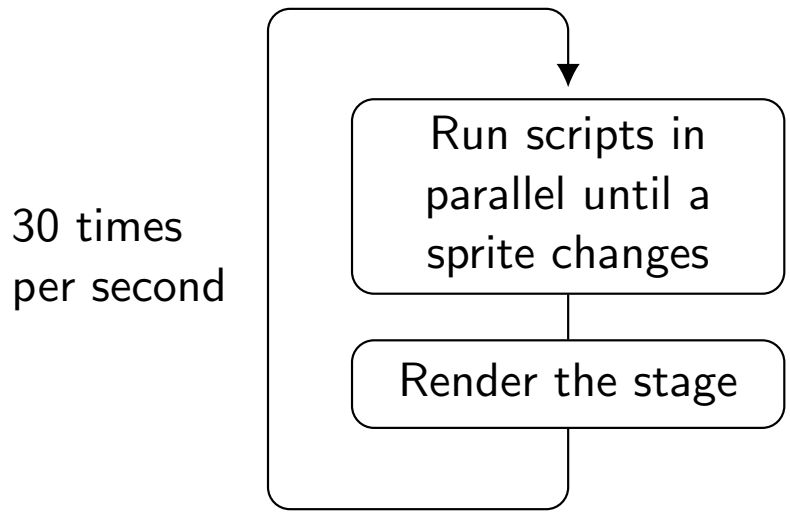


Figure: Scratch step cycle

Why Scratch?

Why Scratch? Scratch's online community

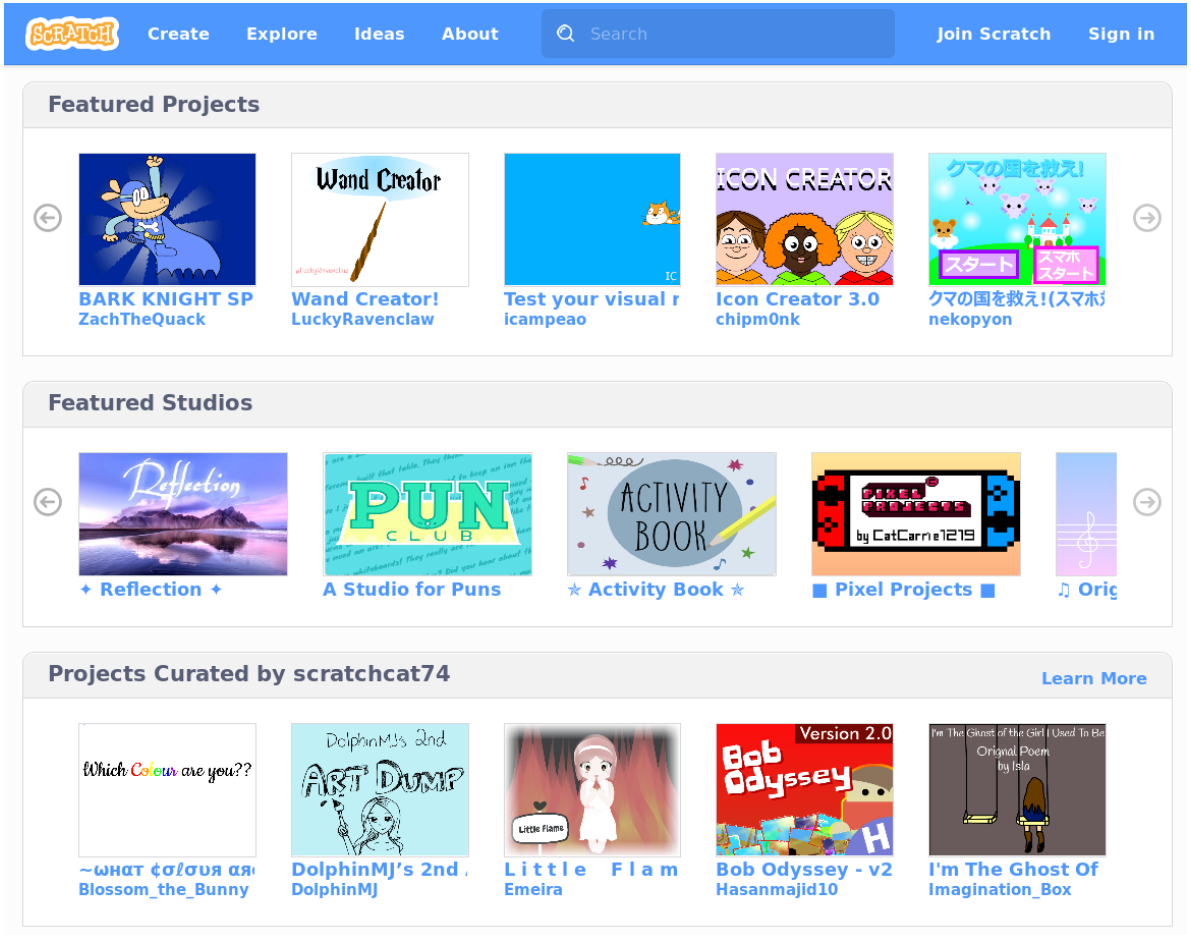


Figure: Scratch's online repository

Why Scratch? Scratch's online community

Figure: Submitted Scratch projects per month

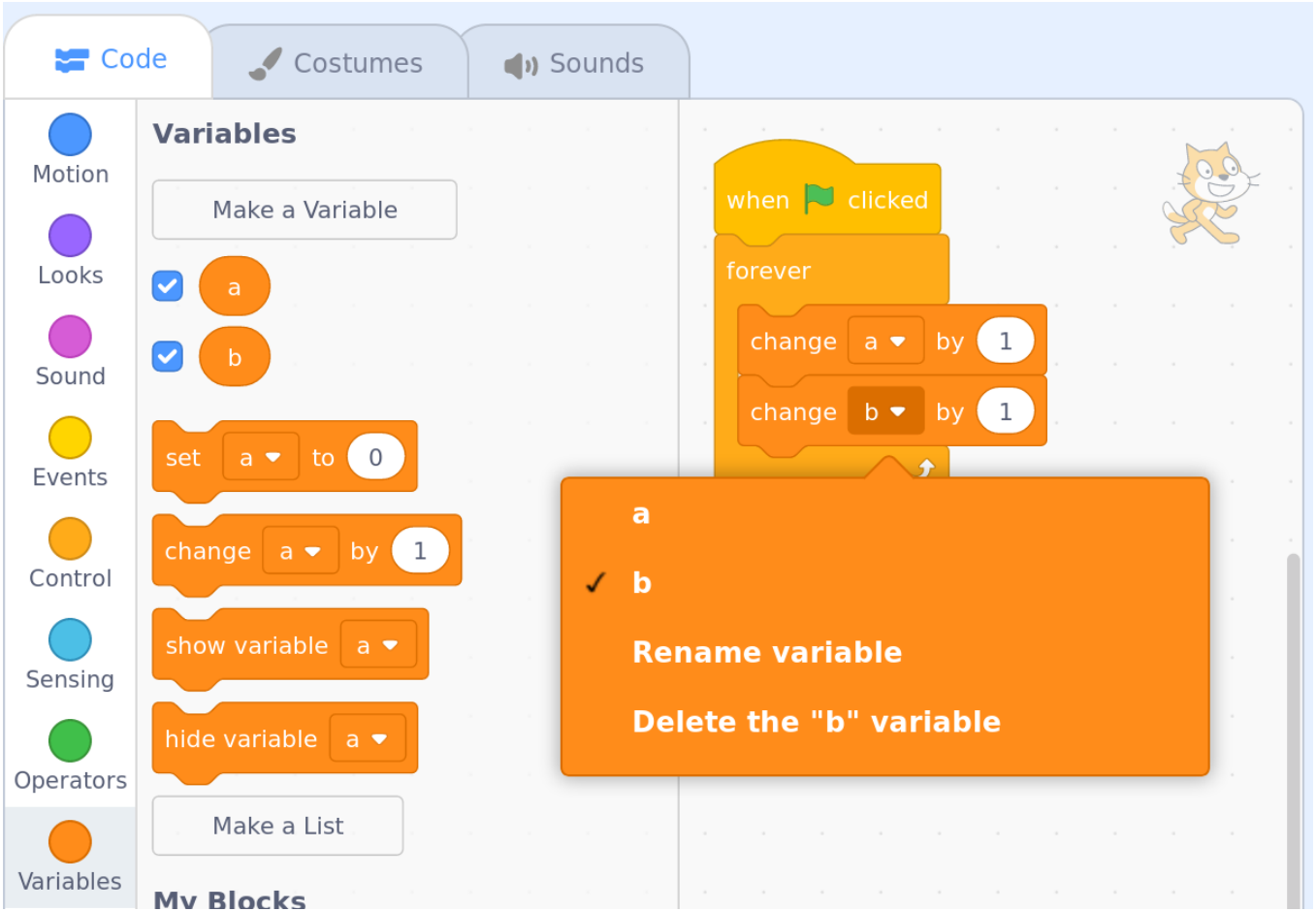
- over 38 million projects shared
- over 36 million users

Why Scratch? Good introduction to programming

Many schools and universities deploy Scratch as a gentle introduction to programming.

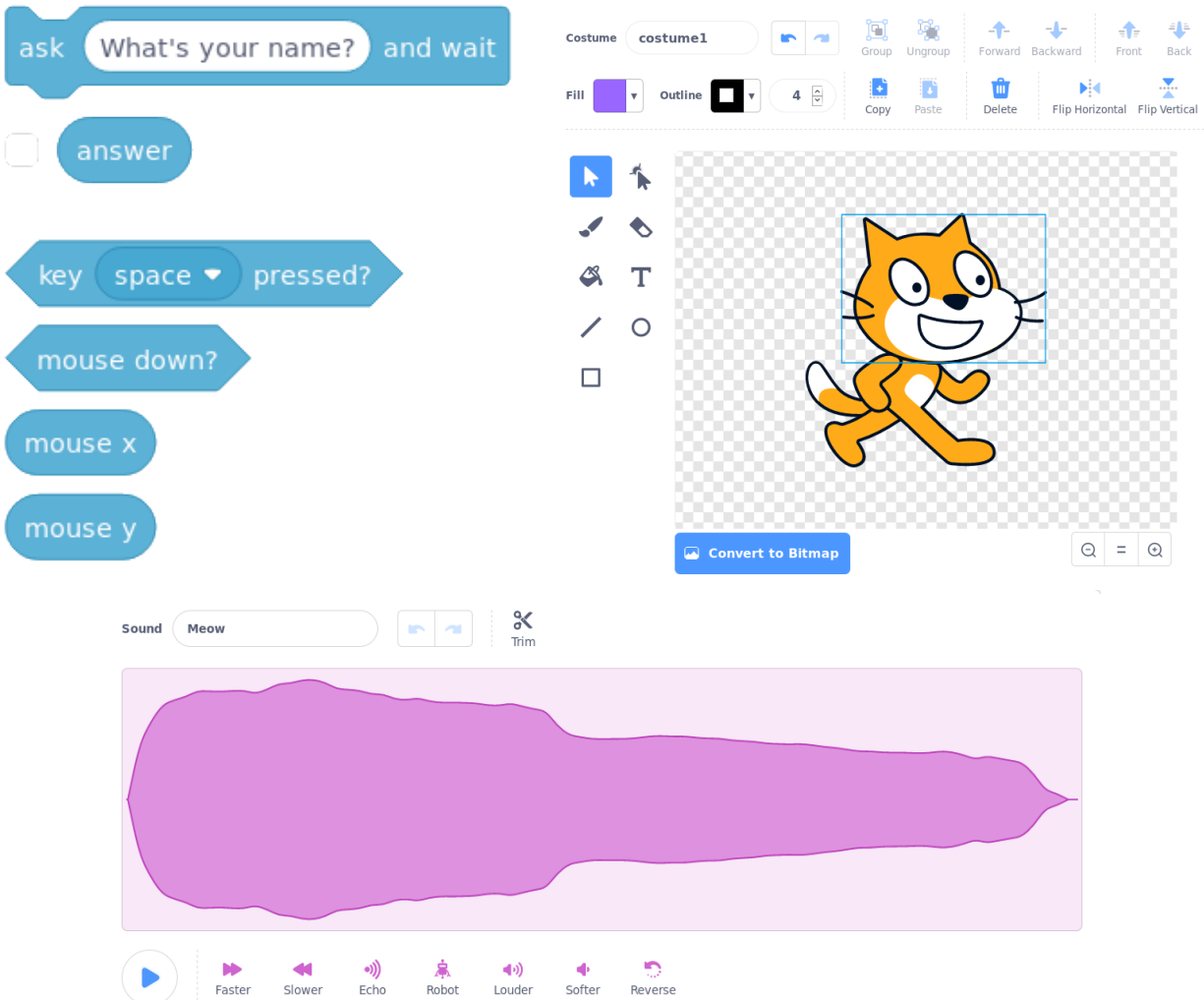
Why Scratch? Good introduction to programming

Intuitive: Block based code system only allows valid code



Why Scratch? Good introduction to programming

Engaging: User interaction, easy integration of graphics and sounds



Why automated testing for Scratch?

Why automated testing for Scratch?

Grading Scratch assignments is very time consuming

- every project has to be opened individually
- programs require large amounts of user interaction

Some courses are attended by a large number of students (> 200), making manual grading infeasible.

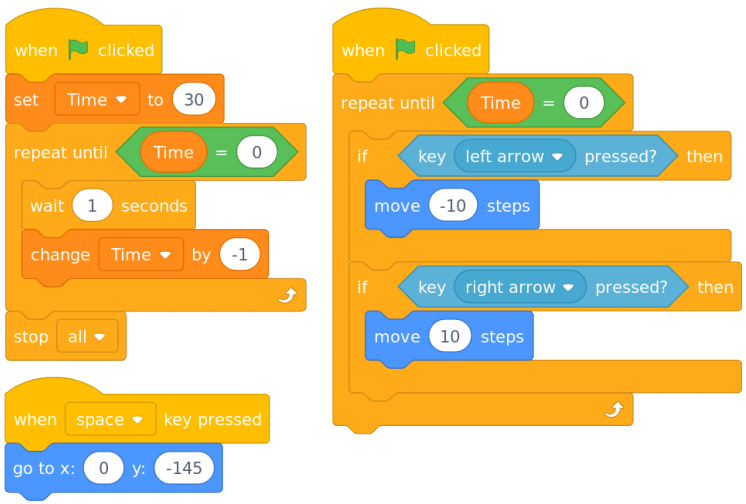
Students can also use automated tests to get feedback for their own implementations.

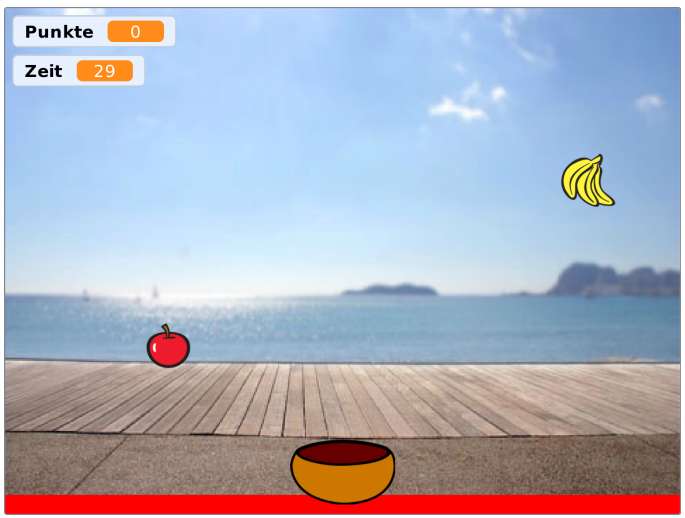
Why is automated testing for Scratch difficult?

Why is automated testing for Scratch difficult?

Usually functional testing is deployed to automatically assess student solution, but this is not straightforward for Scratch

- Scratch is normally only accessible through its GUI
- no functions that take parameters and return a value
- no textual IO, keyboard and mouse input and graphical output



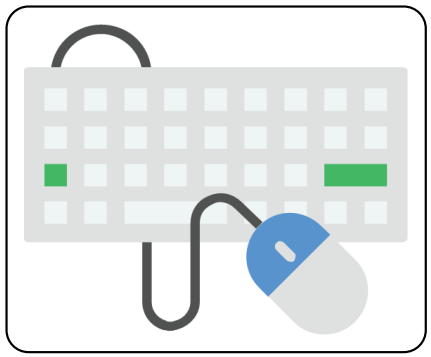


How to test Scratch programs?

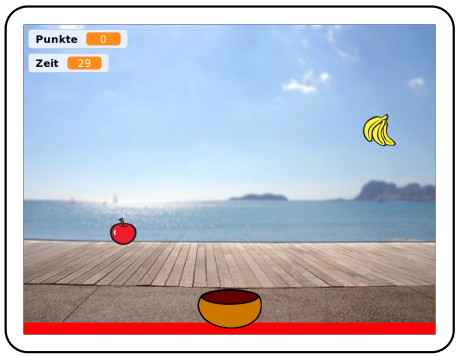
How to test Scratch programs? Automating IO

Approach: Test on a system level by automating Scratch's IO

Input



Output



Input

```
t.inputImmediate({
    device: 'mouse',
    isDown: true,
    x: 50,
    y: 100
});
```

Output

```
sprite.x
sprite.y
sprite.rotation
sprite.sayText
sprite.costume
variable.value
```

How to test Scratch programs? Automating IO

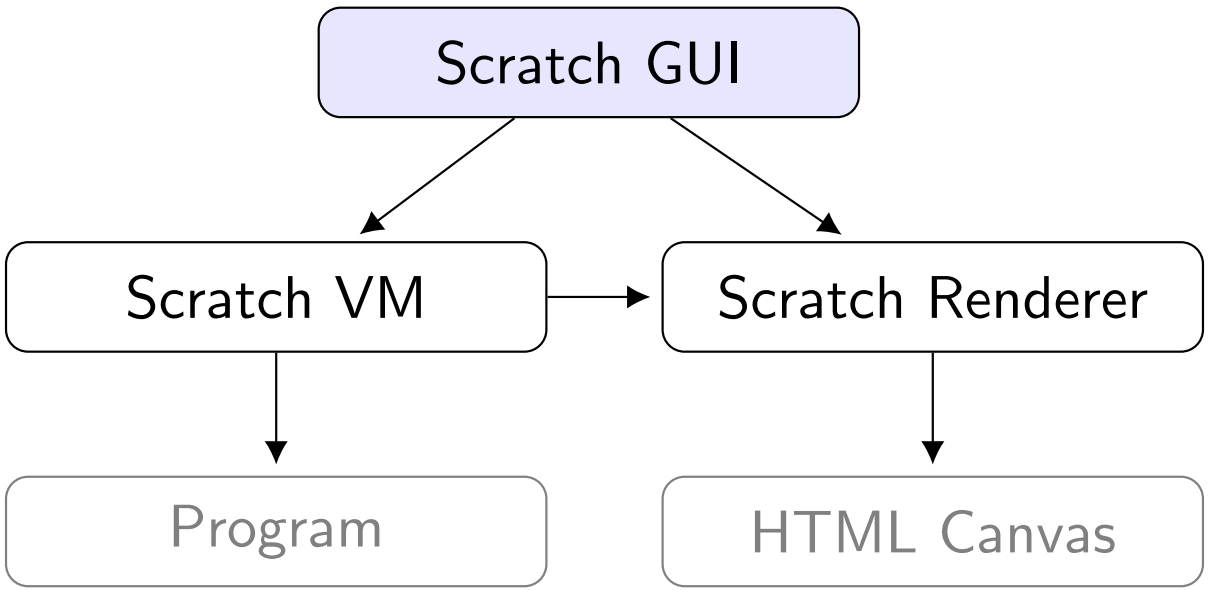


Figure: General architecture of Scratch

How to test Scratch programs? Automating IO

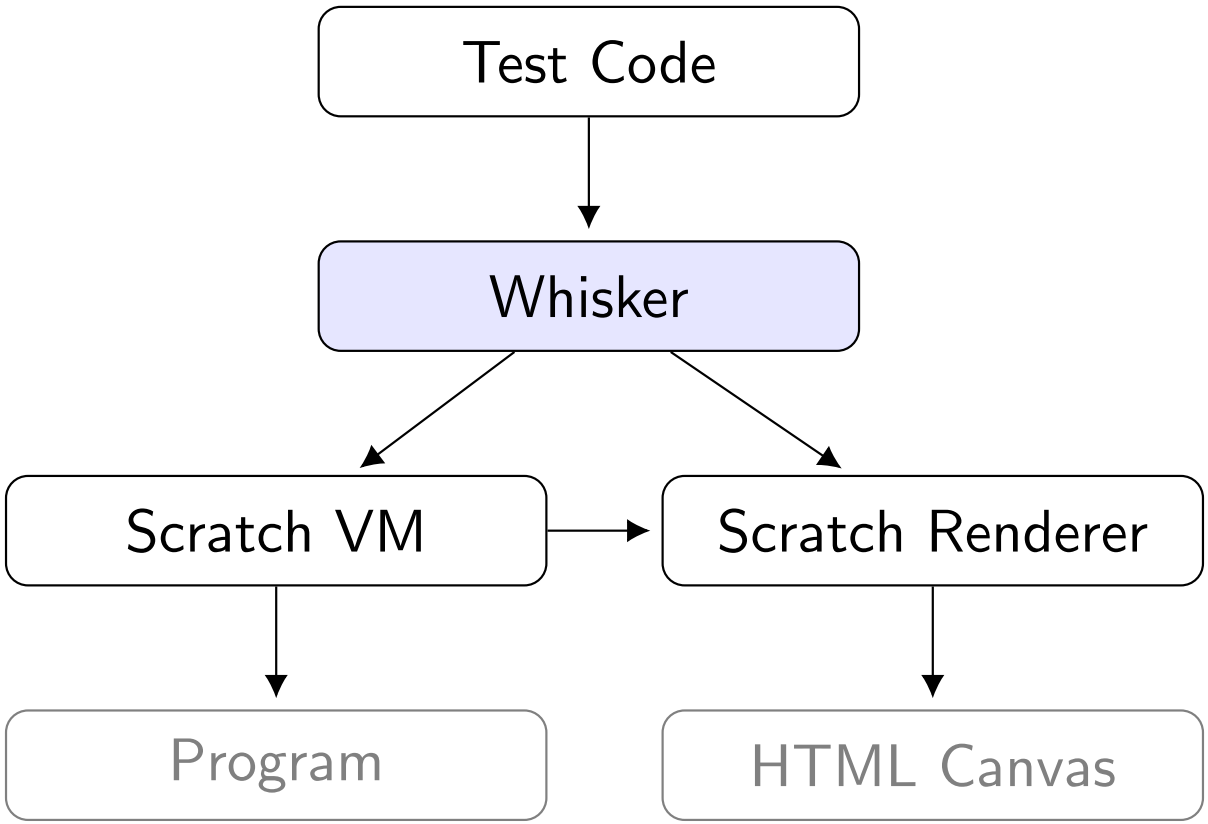
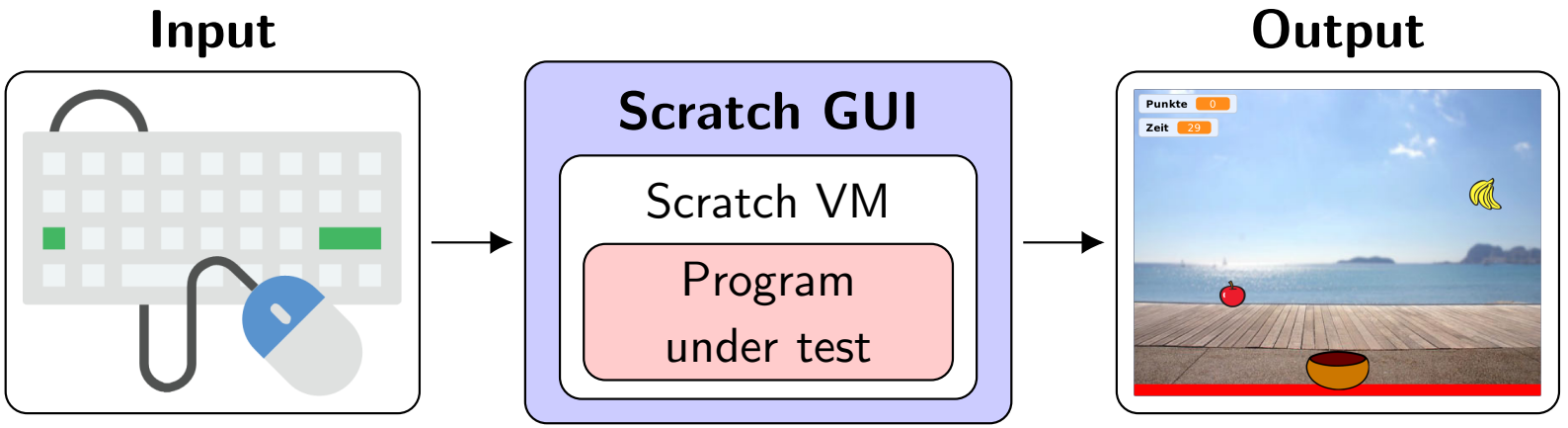
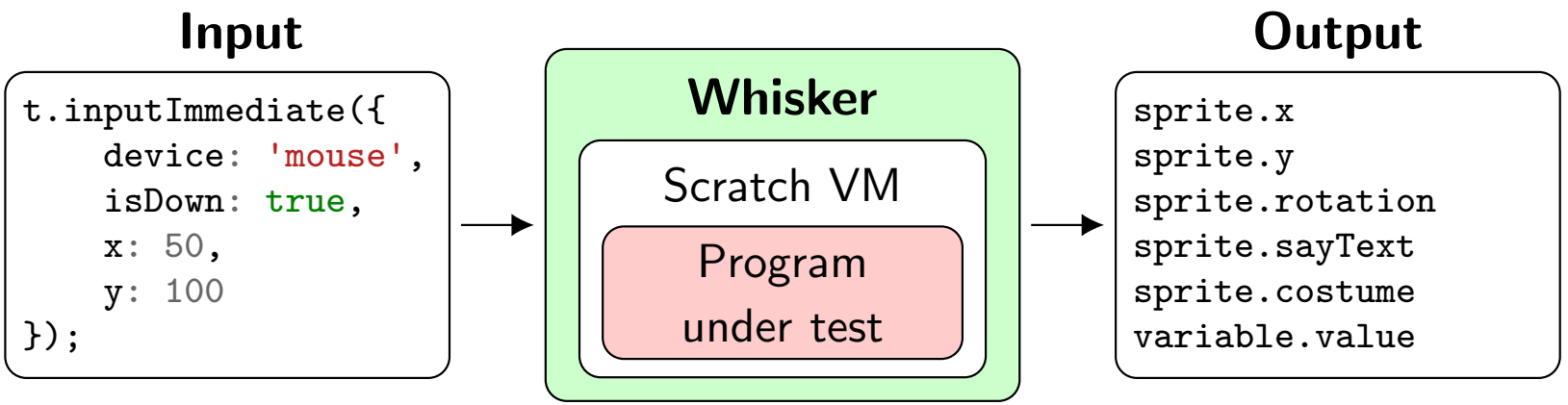


Figure: General architecture of Whisker

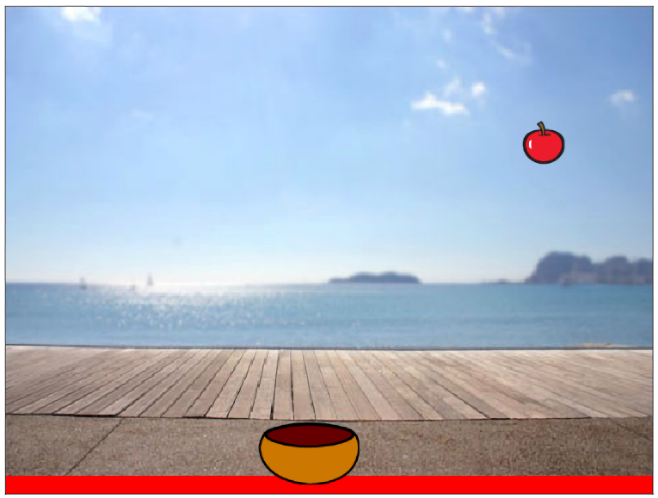
How to test Scratch programs? Automating IO





Whisker

Whisker, Whisker's GUI



Scratch Project

sample.sb3 Browse

Tests

tests.js Browse

Scratch Controls

Reset Run All Tests

Enable Input

Search:

	#	Name	Categories	
+	1	Bowl Initialization Test	initialization, bowl	▶
+	2	Bowl Movement Test	bowl movement, bowl, input	▶
+	3	Bowl Movement Details Test	bowl movement, bowl, input	▶

TAP13

```
# project: sample.sb3
TAP version 13
1..3
ok 1 - Bowl Initialization Test
ok 2 - Bowl Movement Test
not ok 3 - Bowl Movement Details Test
  ---
  severity: fail
  error:
    name: AssertionError
    actual: false
    expected: true
    operator: ok
    message: Bowl must move in steps of size 10.
```

Figure: Whisker's GUI

Whisker, Example Test

```
const test = async function (t) {
    const sprite = t.getSprite('Sprite1');

    await t.runForTime(100);
    let oldX = sprite.x;

    await t.runForTime(1000);

    t.assert.ok(oldX === sprite.x);

    t.inputImmediate({
        device: 'keyboard',
        key: 'right arrow',
        isDown: true
    });

    await t.runForTime(1000);

    t.assert.ok(oldX < sprite.x);
}
```

Whisker, Accessing sprites and variables

```
const test = async function (t) {
    const sprite =

    await t.runForT
    let oldX = spri

    await t.runForT

    t.assert.ok(old

    t.inputImmediat
        device: 'ke
        key: 'right
        isDown: tru
    });

    await t.runForT

    t.assert.ok(old
}
```

```
t.getSprite('Sprite1');
t.getSprites(sprite => sprite.x > 100);
sprite.getClones();
t.getStage();

stage.getVariable('my variable');
stage.getVariables();
sprite.getList('my list');
sprite.getLists();

sprite.x;
sprite.old.x;
variable.value;

sprite.isOriginal();
sprite.isTouchingEdge();
sprite.isTouchingSprite(otherSprite);
```

Whisker, Running the program

```
const test = async function (t) {
    const sprite = t.getSprite('Sprite1');

    await t.runForTime(100);
    let oldX = sprite.x;

    await t.runForT
    t.assert.ok(old

    t.inputImmediat
        device: 'ke
        key: 'right
        isDown: tru
    });

    await t.runForTime(1000);

    t.assert.ok(oldX < sprite.x);
}
```

```
await t.runForTime(1000);
await t.runForSteps(30);
await t.runUntil(() => a > b, 1000);

t.getRunTimeElapsed();
t.getTotalTimeElapsed();

t.greenFlag();
```

Whisker, Simulating Inputs

```
t.inputImmediate({
    device: 'keyboard',
    key: 'right arrow',
    isDown: true,
    duration: 100
});
t.addInput(1000, {
    device: 'mouse',
    x: 100,
    y: 200,
    isDown: true
});
t.addInput(2000, {
    device: 'text',
    text: 'some answer'
});

t.getMousePos();
t.isMouseDown();
t.isKeyDown('space');
```

Whisker, Accessing Output

```
const sprite = t.getSprite('Sprite1');
const sprites = t.getSprites(sprite => sprite.x > 100);
const clones = sprite.getClones();
const stage = t.getStage();

const variable = stage.getVariable('my variable');
const variables = stage.getVariables();
const list = sprite.getList('my list');
const lists = sprite.getLists();

sprite.x;
sprite.old.x;
variable.value;

sprite.isOriginal();
sprite.isTouchingEdge();
sprite.isTouchingSprite(otherSprite);
```

Whisker, Running the program

```
await t.runForTime(1000);
await t.runForSteps(30);
await t.runUntil(() => a > b, 1000);

t.getRunTimeElapsed();
t.getTotalTimeElapsed();

t.greenFlag();
```

Whisker, Callbacks

```
const callback = t.addCallback(() => {
    if (sprite.x > 100) {
        t.inputImmediate({ device: 'mouse', isDown: true });
    } else if (sprite.x < 0) {
        t.cancelRun();
    }
});

t.addCallback(() => someList.push(sprite.x), true);

callback.disable();
callback.enable();
callback.isActive();
```

Whisker, Constraints

```
t.onConstraintFailure('fail');
t.onConstraintFailure('nothing');

const constraint = t.addConstraint(() => {
    t.assert.ok(sprite.visible === true,
        'Sprite must always be visible.');
});

constraint.disable();
constraint.enable();
constraint.isActive();
```

Whisker, Input Generation

```
t.setRandomInputInterval(150);

t.registerRandomInputs([
  { device: 'keyboard', key: 'left arrow', duration: [50, 100] },
  { device: 'keyboard', key: 'right arrow', duration: [50, 100] },
  { device: 'mouse', x: [-100, 100], y: [-100, 100], weight: 0.5 }
]);

t.detectRandomInputs({ duration: [50, 100] });
```