Team Members

Jacob Lopez – jlopez70
Daniel Moon – dmoon9
Maria Movsheva – mmovshe1

Project Description

Our subject software is JBox2D, an open-sourced 2D physics engine written entirely in Java. It is a Java port of the C++ physics engine Box2D. We tested the Collision class within the Collision module along with the entirety of the Shapes module. We wrote all our unit tests in Java using JUnit5. We also analyzed our statement and branch coverage using Jacoco. Finally, mutation analysis was performed using PIT.

Directory Description

Here is a description of our final submission directory.

- **Readme.pdf**: This file. Describes the project, directory, and achieved results.
- **Project**: A Maven project which is a git fork of the original SUT. All of our tests and code can be found in `Project/jbox2d-library/src/test/java/SoftwareTestingUnitTests`. Some of the `pom.xml` files were also updated to properly compile newer dependency versions. The rest of the folder came from the original SUT.
- **PIT_Compatible_Project**: This is a modified copy of `Project`. Specifically, some of our unit tests have been commented out. Pittest only runs for green test suites so we had to comment out any failing tests.
- **Reports**: Contains the generated Jacoco and PIT reports.
    - jacoco: Contains the generated jacoco report from `Project`. The report can be viewed by opening `Reports/jacoco/index.html` on a browser.
    - pit-reports: Contains the generated PIT report from `PIT_Compatible_Project`.
- **Presentation**: Contains our final presentation in powerpoint format.

How to Run

There are three primary things that can be executed. Firstly, you can run the tests in a standalone fashion. Secondly, you can run the tests and generate a Jacoco report. Finally, you can run the tests and generate a Pitest report. The instructions a different based on whether you are using a Windows or MacOS and are provided before

Initial Setup and Dependencies

This code runs on Java 11, and maven must be installed since this is a maven project.

Instructions for MacOS

How to run tests

1. In a terminal, navigate to `Project/jbox2d-library`
2. Run `mvn test`

How to generate Jacoco report

1. In a terminal, navigate to `Project/jbox2d-library`
2. Run `mvn clean test -Dmaven.test.failure.ignore=true`

How to generate PIT report

1. In a terminal, navigate to `Project/jbox2d-library`
2. Run `mvn test-compile org.pitest:pitest-maven:mutationCoverage`

Instructions for Windows

How to run tests

3. In a terminal, navigate to `Project/jbox2d-library`
4. Run `mvn test`

How to generate Jacoco report

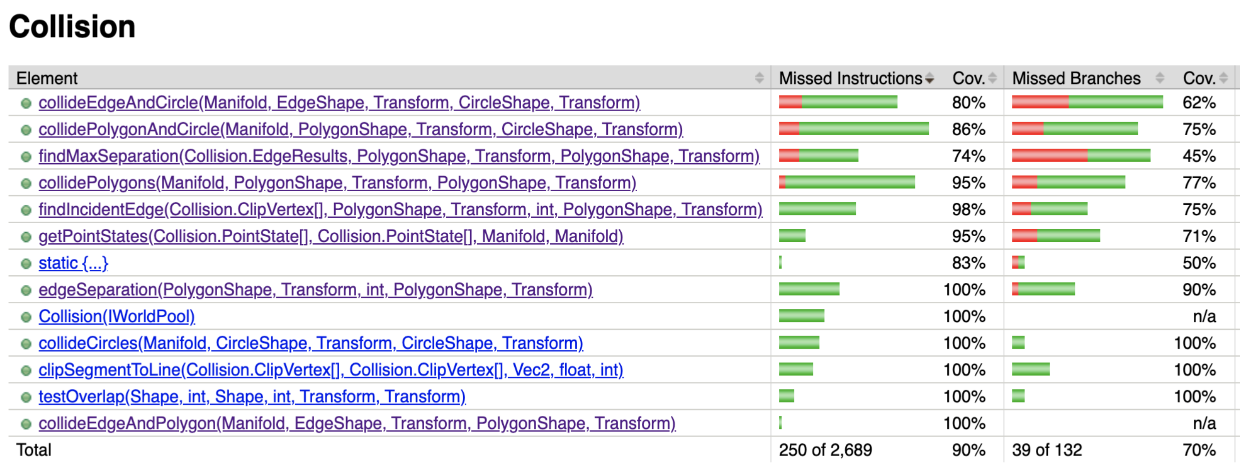
3. In a terminal, navigate to `Project/jbox2d-library`
4. Run `mvn clean test -D”maven.test.failure.ignore”=”true`”

How to generate PIT report

3. In a terminal, navigate to `Project/jbox2d-library`
4. Run `mvn test-compile org.pitest:pitest-maven:mutationCoverage`

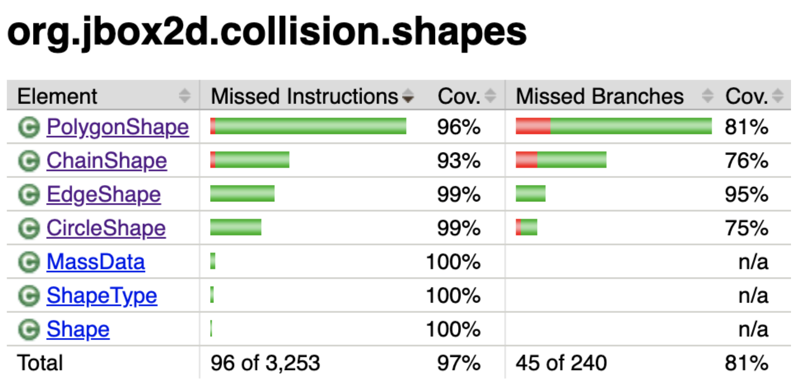
Results

Coverage testing for shapes and collision class both resulted in over 90% statement coverage and over 70% in branch coverage. Several of the branches were either unreachable due to the structure of the code (asserts that could never pass, if statements that were duplicated and returned if satisfied, etc.). Also, many of the branches in collision were very difficult to reach, and while many of them may be reachable with very specific inputs, we did not find those sets of inputs to cover all the possible branches. Hence, we plateaued at around 70% for collision class.



Collision

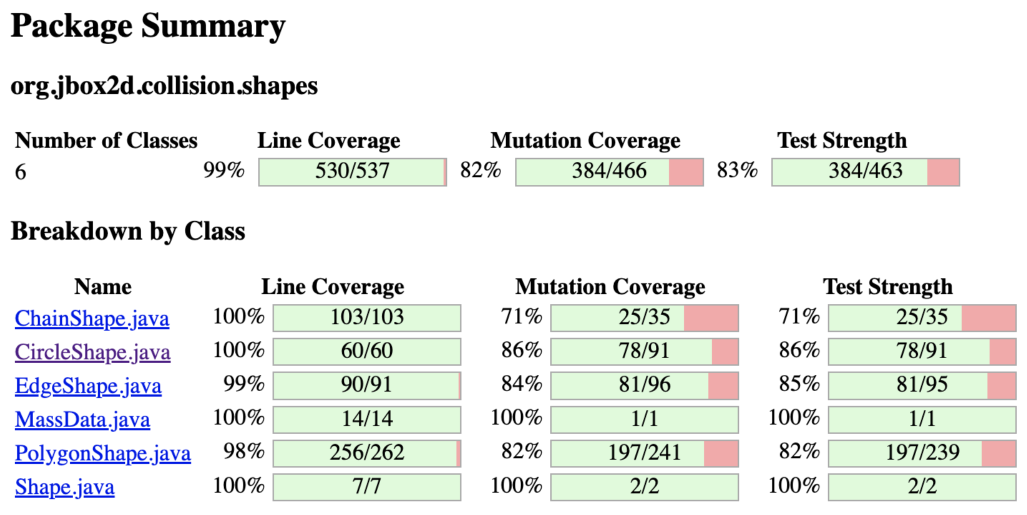
Element	Missed Instructions	Cov.	Missed Branches	Cov.
collideEdgeAndCircle(Manifold, EdgeShape, Transform, CircleShape, Transform)		80%		62%
collidePolygonAndCircle(Manifold, PolygonShape, Transform, CircleShape, Transform)		86%		75%
findMaxSeparation(Collision.EdgeResults, PolygonShape, Transform, PolygonShape, Transform)		74%		45%
collidePolygons(Manifold, PolygonShape, Transform, PolygonShape, Transform)		95%		77%
findIncidentEdge(Collision.ClipVertex[], PolygonShape, Transform, int, PolygonShape, Transform)		98%		75%
getPointStates(Collision.PointState[], Collision.PointState[], Manifold, Manifold)		95%		71%
static {...}		83%		50%
edgeSeparation(PolygonShape, Transform, int, PolygonShape, Transform)		100%		90%
Collision(IWorldPool)		100%		n/a
collideCircles(Manifold, CircleShape, Transform, CircleShape, Transform)		100%		100%
clipSegmentToLine(Collision.ClipVertex[], Collision.ClipVertex[], Vec2, float, int)		100%		100%
testOverlap(Shape, int, Shape, int, Transform, Transform)		100%		100%
collideEdgeAndPolygon(Manifold, EdgeShape, Transform, PolygonShape, Transform)		100%		n/a
Total	250 of 2,689	90%	39 of 132	70%



org.jbox2d.collision.shapes

Element	Missed Instructions	Cov.	Missed Branches	Cov.
PolygonShape		96%		81%
ChainShape		93%		76%
EdgeShape		99%		95%
CircleShape		99%		75%
MassData		100%		n/a
ShapeType		100%		n/a
Shape		100%		n/a
Total	96 of 3,253	97%	45 of 240	81%

Mutation testing for the shape classes reached over 80%. The mutation coverage for some classes could not reach 100% due to some lines being outright unreachable, and other mutations being equivalent mutants which were also impossible to kill. Aside from these cases, most of the mutants that were killable were killed in all shape classes. Chainshapes had a lower mutant kill rate since some asserts in that class would always evaluate to false and return.



Package Summary

org.jbox2d.collision.shapes

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
6	99%	530/537	82%	384/466	83%	384/463

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
ChainShape.java	100%	103/103	71%	25/35	71%	25/35
CircleShape.java	100%	60/60	86%	78/91	86%	78/91
EdgeShape.java	99%	90/91	84%	81/96	85%	81/95
MassData.java	100%	14/14	100%	1/1	100%	1/1
PolygonShape.java	98%	256/262	82%	197/241	82%	197/239
Shape.java	100%	7/7	100%	2/2	100%	2/2