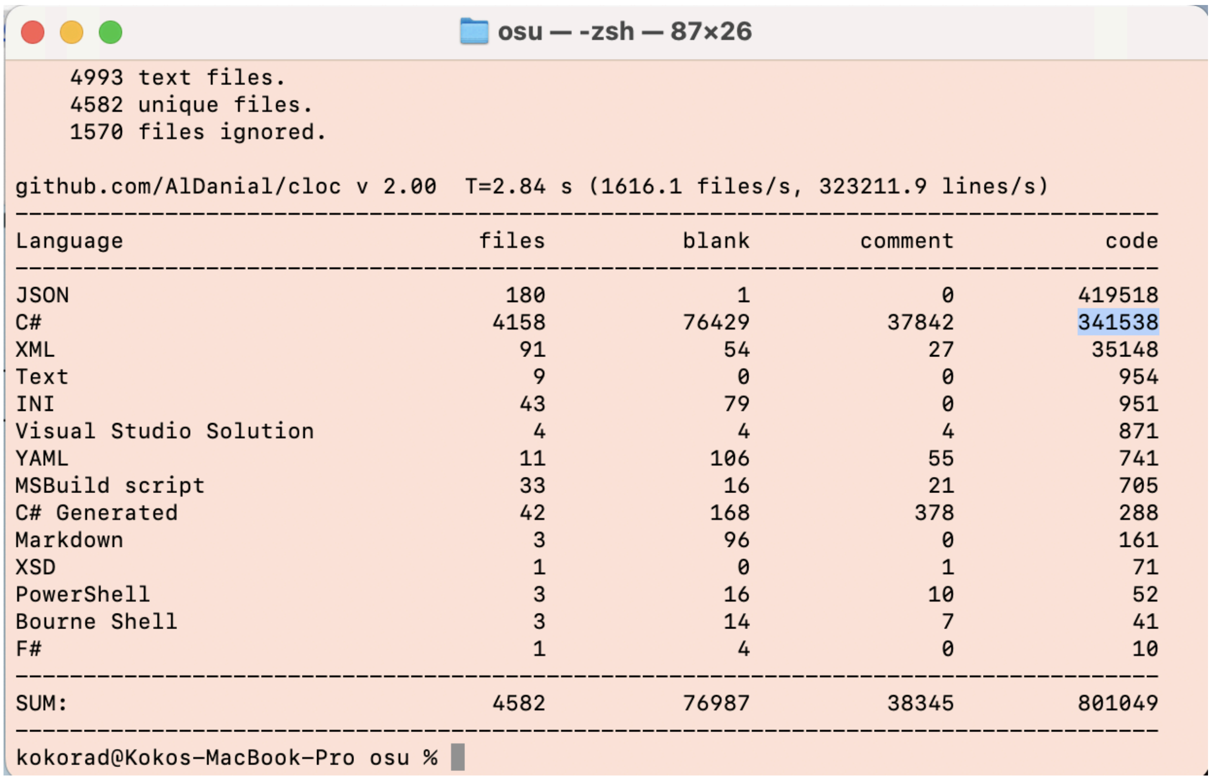
Report for Assignment 1

- ## **Project chosen:**

Name: osu

URL: https://github.com/ppy/osu

Number of lines of code and the tool used to count it: 341.538 KLOC, we used “cloc”:



osu — -zsh — 87×26

```
     4993 text files.
     4582 unique files.
     1570 files ignored.

github.com/AlDanial/cloc v 2.00  T=2.84 s (1616.1 files/s, 323211.9 lines/s)
-------------------------------------------------------------------------------------
Language                             files          blank        comment           code
-------------------------------------------------------------------------------------
JSON                                   180              1              0         419518
C#                                    4158          76429          37842         341538
XML                                     91             54             27          35148
Text                                     9              0              0            954
INI                                     43             79              0            951
Visual Studio Solution                   4              4              4            871
YAML                                    11            106             55            741
MSBuild script                          33             16             21            705
C# Generated                            42            168            378            288
Markdown                                 3             96              0            161
XSD                                      1              0              1             71
PowerShell                               3             16             10             52
Bourne Shell                             3             14              7             41
F#                                       1              4              0             10
-------------------------------------------------------------------------------------
SUM:                                  4582          76987          38345         801049
-------------------------------------------------------------------------------------
kokorad@Kokos-MacBook-Pro osu %
```

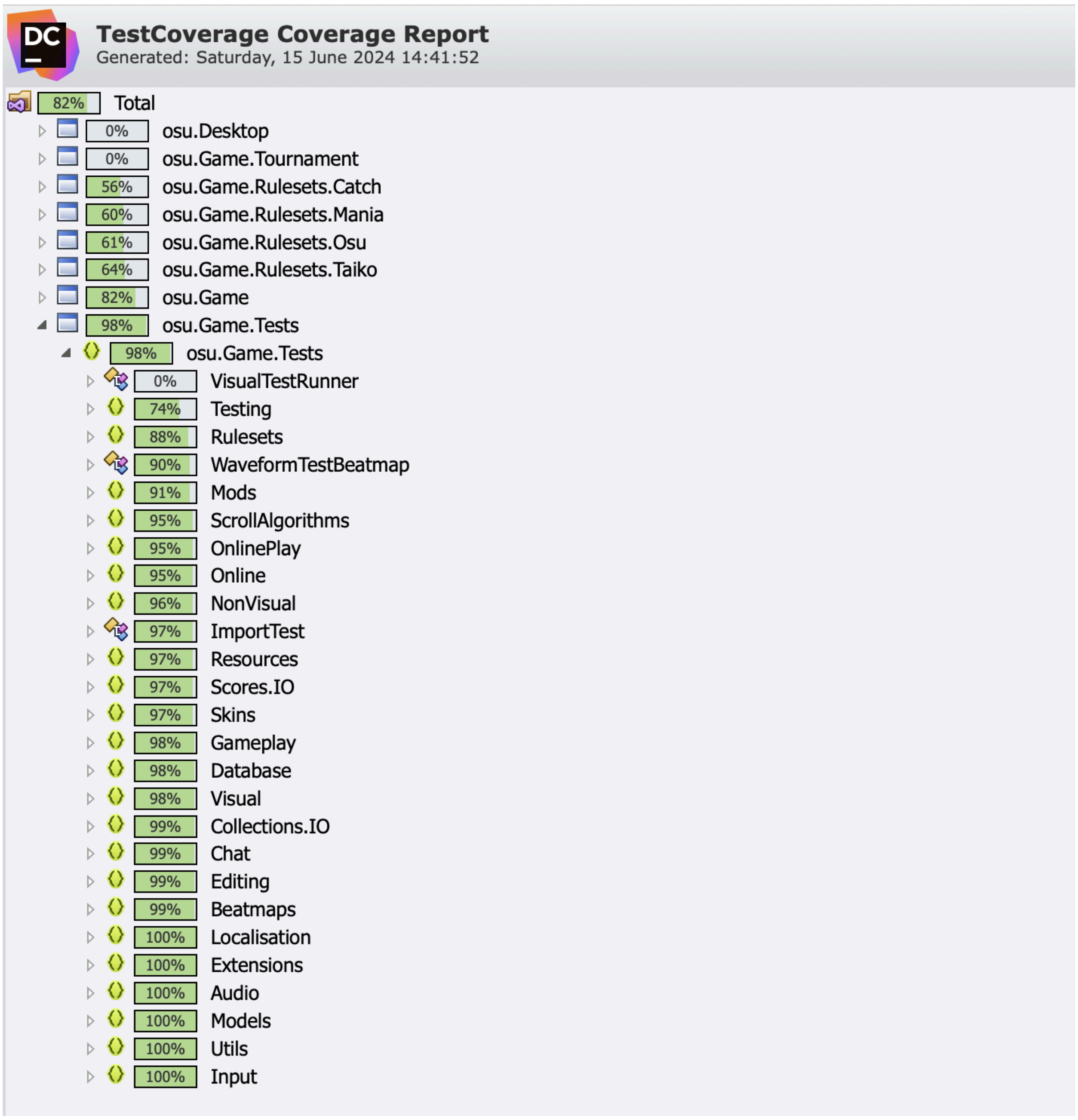
Programming language: C#

- ## **Coverage measurement**

Existing tool

- The name of the existing tool that was executed and how it was executed: ***dotCover***
- Steps: .JetRider -> View -> Tool Windows -> Unit Test Coverage

- The coverage results provided by the existing tool with a screenshot:



TestCoverage Coverage Report
Generated: Saturday, 15 June 2024 14:41:52

```
82%  Total
  0%   osu.Desktop
  0%   osu.Game.Tournament
  56%  osu.Game.Rulesets.Catch
  60%  osu.Game.Rulesets.Mania
  61%  osu.Game.Rulesets.Osu
  64%  osu.Game.Rulesets.Taiko
  82%  osu.Game
  98%  osu.Game.Tests
    98%  osu.Game.Tests
      0%    VisualTestRunner
      74%   Testing
      88%   Rulesets
      90%   WaveformTestBeatmap
      91%   Mods
      95%   ScrollAlgorithms
      95%   OnlinePlay
      95%   Online
      96%   NonVisual
      97%   ImportTest
      97%   Resources
      97%   Scores.IO
      97%   Skins
      98%   Gameplay
      98%   Database
      98%   Visual
      99%   Collections.IO
      99%   Chat
      99%   Editing
      99%   Beatmaps
      100%  Localisation
      100%  Extensions
      100%  Audio
      100%  Models
      100%  Utils
      100%  Input
```

- **Our Own Coverage Tool:**

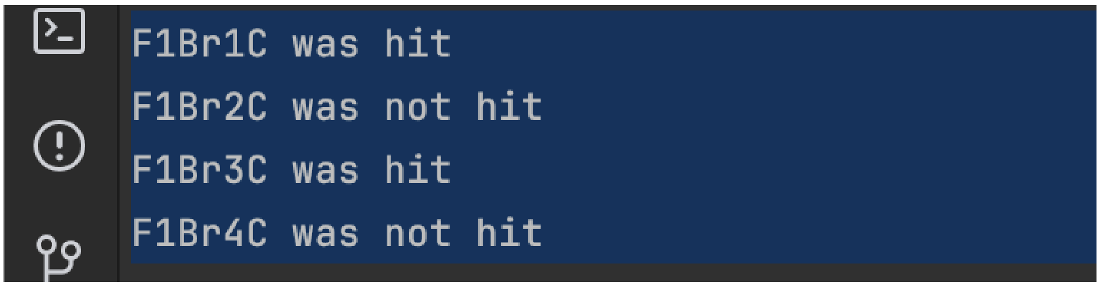
Group member 1: *Cosmina-Andreea Radus*

Function 1 name: getClampedQuad(Axes clampAxes, Vector2 topLeft, Vector2 size)

- *Path: osu.Game/Graphics/Backgrounds/TrianglesV2.cs*

- A link to a commit made in the forked repository that shows the instrumented code to gather coverage measurements:
https://github.com/marwat16/SEP_group122_osu/blob/Cosmina/osu.Game/Graphics/Backgrounds/TrianglesV2.cs

- A screenshot of the coverage results output by the instrumentation:



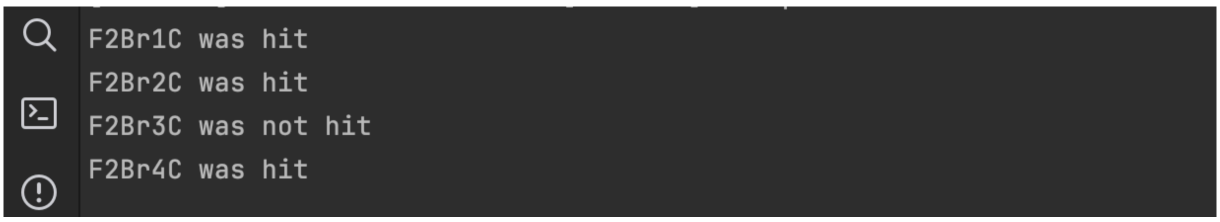
```
F1Br1C was hit
F1Br2C was not hit
F1Br3C was hit
F1Br4C was not hit
```

Function 2 name: downHeap(T[] keys, int i, int n, int lo, IComparer<T> comparer)

- *Path: osu.Game.Rulesets.Mania/MathUtils/LegacySortHelper.cs*

- A link to a commit made in the forked repository that shows the instrumented code to gather coverage measurements:
https://github.com/marwat16/SEP_group122_osu/blob/Cosmina/osu.Game.Rulesets.Mania/MathUtils/LegacySortHelper.cs

- A screenshot of the coverage results output by the instrumentation:



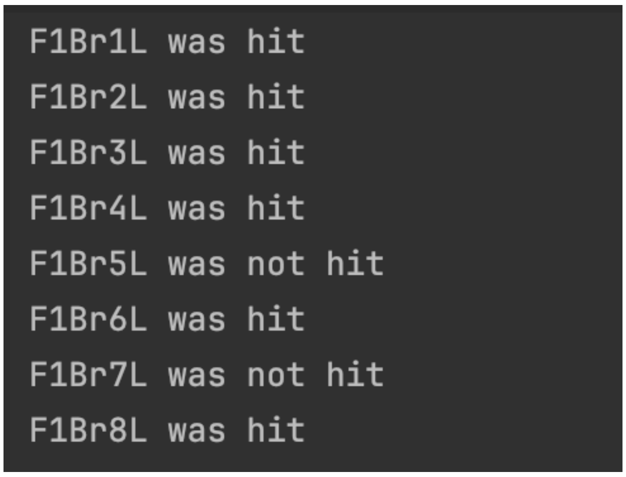
```
F2Br1C was hit
F2Br2C was hit
F2Br3C was not hit
F2Br4C was hit
```

Group member 2: *Lama Abboud*

Function 1: moveSelectionInBounds()

- A link to a commit made in the forked repository that shows the instrumented code to gather coverage measurements: https://github.com/marwat16/SEP_group122_osu/blob/master/osu.Game.Rulesets.Osu/Edit/OsuSelectionScaleHandler.cs

- A screenshot of the coverage results output by the instrumentation:

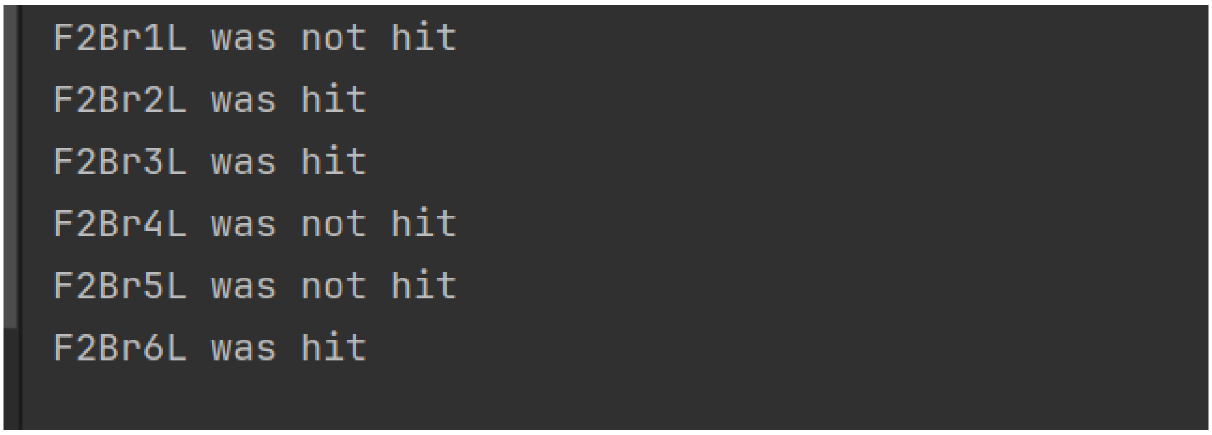


```
F1Br1L was hit
F1Br2L was hit
F1Br3L was hit
F1Br4L was hit
F1Br5L was not hit
F1Br6L was hit
F1Br7L was not hit
F1Br8L was hit
```

Function 2: simulateHit(HitObject hitObject, ref LegacyScoreAttributes attributes)

- A link to a commit made in the forked repository that shows the instrumented code to gather coverage measurements: https://github.com/marwat16/SEP_group122_osu/blob/master/osu.Game.Rulesets.Catch/Difficulty/CatchLegacyScoreSimulator.cs

- A screenshot of the coverage results output by the instrumentation:



```
F2Br1L was not hit
F2Br2L was hit
F2Br3L was hit
F2Br4L was not hit
F2Br5L was not hit
F2Br6L was hit
```

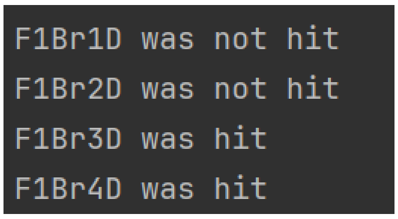
Group member 3: *Daphne Lefevre*

Function 1: Equals(Background other)

- Path: osu.Game/Graphics/Backgrounds/Background.cs
- A link to a commit made in the forked repository that shows the instrumented code to gather coverage measurements:

https://github.com/marwat16/SEP_group122_osu/blob/Daphn%C3%A9/osu.Game/Graphics/Backgrounds/Background.cs

- A screenshot of the coverage results output by the instrumentation:



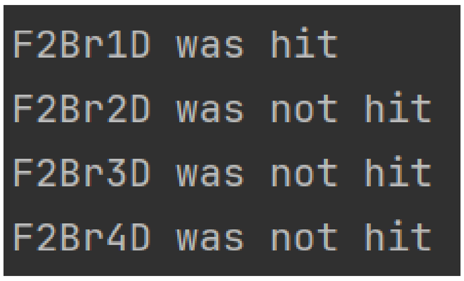
```
F1Br1D was not hit
F1Br2D was not hit
F1Br3D was hit
F1Br4D was hit
```

Function 2: FromLegacy(LegacyReplayFrame currentFrame, IBeatmap beatmap, ReplayFrame? lastFrame = null)

- Path: osu.Game.Rulesets.Catch/Replays/CatchReplayFrame.cs
- A link to a commit made in the forked repository that shows the instrumented code to gather coverage measurements:

https://github.com/marwat16/SEP_group122_osu/blob/Daphné/osu.Game.Rulesets.Catch/Replays/CatchReplayFrame.cs

- A screenshot of the coverage results output by the instrumentation:



```
F2Br1D was hit
F2Br2D was not hit
F2Br3D was not hit
F2Br4D was not hit
```

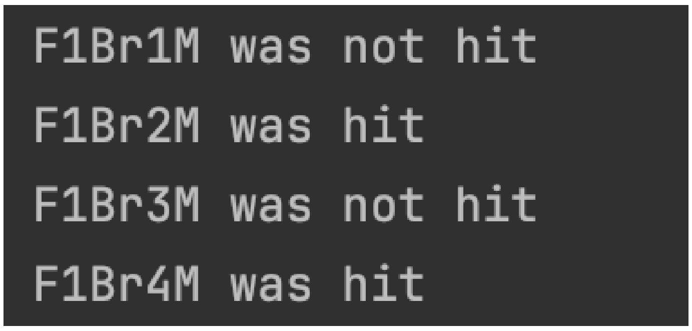
Group member 4: *Marwa Tinmiouli*

Function 1: AddPoint(Vector2 start, Vector2 end, Vector2 hitPoint, float radius)

- A link to a commit made in the forked repository that shows the instrumented code to gather coverage measurements:

https://github.com/marwat16/SEP_group122_osu/blob/master/osu.Game.Rulesets.Osu/Statistics/AccuracyHeatmap.cs

- A screenshot of the coverage results output by the instrumentation:



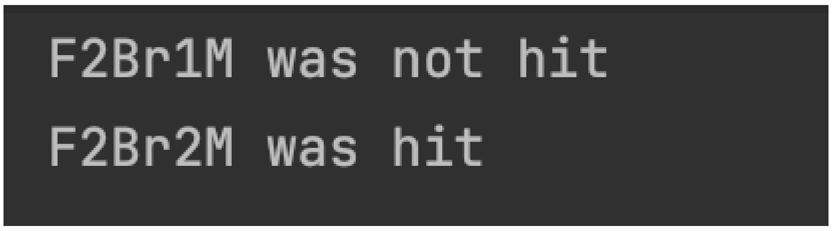
```
F1Br1M was not hit
F1Br2M was hit
F1Br3M was not hit
F1Br4M was hit
```

Function 2: onDirectionChanged(ValueChangedEvent<ScrollingDirection> direction)

- A link to a commit made in the forked repository that shows the instrumented code to gather coverage measurements:

https://github.com/marwat16/SEP_group122_osu/blob/master/osu.Game.Rulesets.Mania/UI/DefaultHitExplosion.cs

- A screenshot of the coverage results output by the instrumentation:



```
F2Br1M was not hit
F2Br2M was hit
```

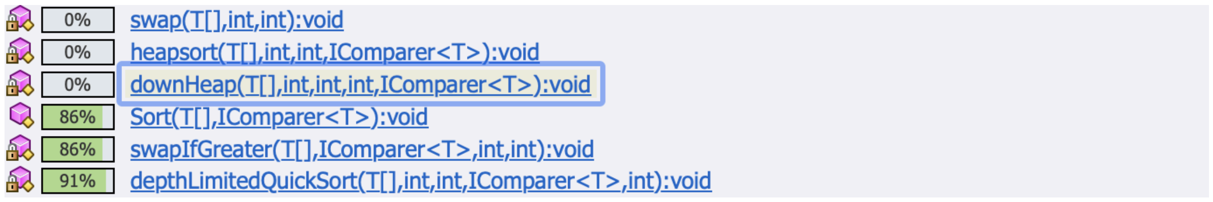
- <u>**Coverage improvement**</u>

Group member 1: *Cosmina-Andreea Radus*

Test 1: <u>TestSceneGetClampedQuad</u>

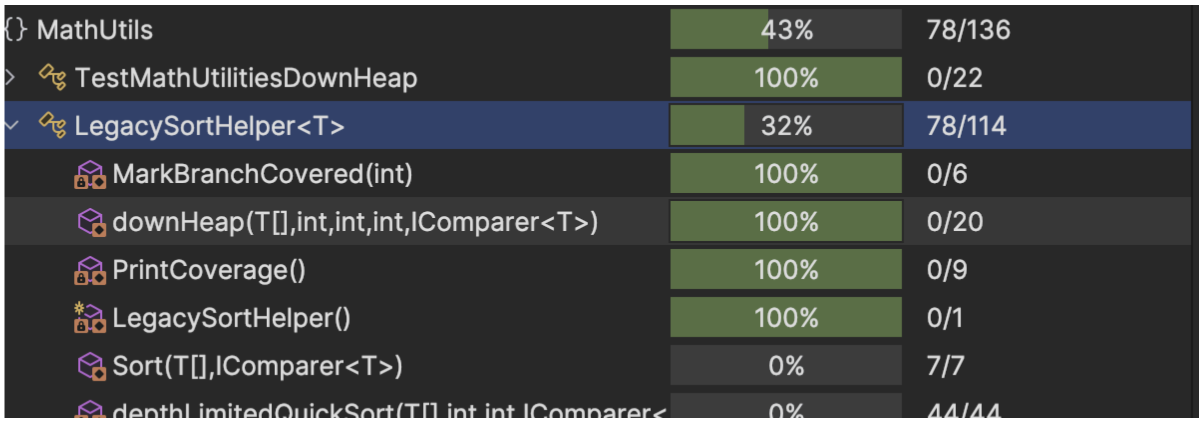
- A link to a commit on the forked repository of the new test: https://github.com/marwat16/SEP_group122_osu/blob/Cosmina/osu.Game.Tests/Visual/Background/TestSceneGetClampedQuad.cs

- A screenshot of the old coverage results:



0%	swap(T[],int,int):void
0%	heapsort(T[],int,int,IComparer<T>):void
0%	downHeap(T[],int,int,int,IComparer<T>):void
86%	Sort(T[],IComparer<T>):void
86%	swapIfGreater(T[],IComparer<T>,int,int):void
91%	depthLimitedQuickSort(T[],int,int,IComparer<T>,int):void

- A screenshot of the new coverage results:



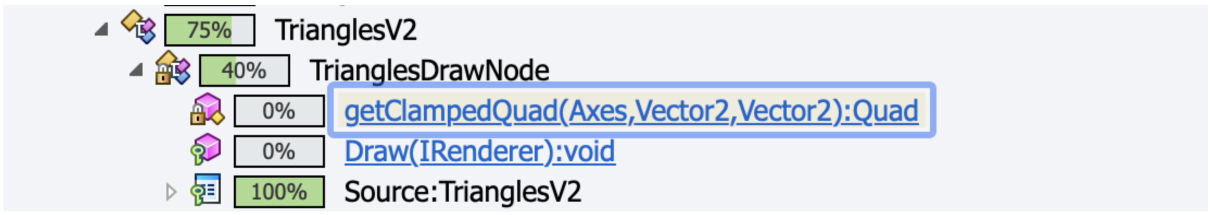
{} MathUtils	43%	78/136
TestMathUtilitiesDownHeap	100%	0/22
LegacySortHelper<T>	32%	78/114
MarkBranchCovered(int)	100%	0/6
downHeap(T[],int,int,int,IComparer<T>)	100%	0/20
PrintCoverage()	100%	0/9
LegacySortHelper()	100%	0/1
Sort(T[],IComparer<T>)	0%	7/7
depthLimitedQuickSort(T[],int,int,IComparer<	0%	44/44

The coverage improvements to 100% (both branch and statement coverage) demonstrate thorough testing of the `getClampedQuad` method.
The tests effectively cover all critical paths, ensuring that all possible outcomes of conditional branches and logic are verified.

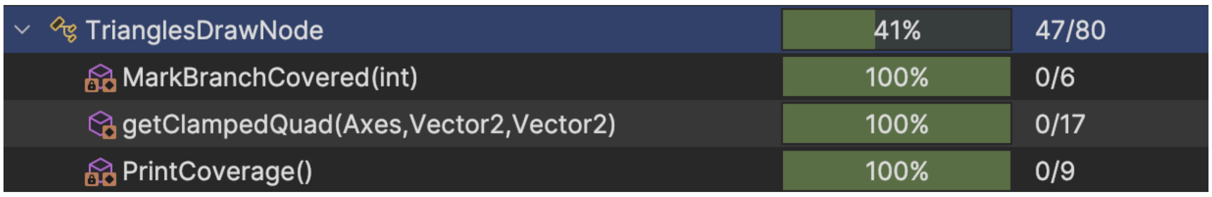
Test 2: <u>TestMathUtilitiesDownHeap</u>

- A link to a commit on the forked repository of the new test:
https://github.com/marwat16/SEP_group122_osu/blob/Cosmina/osu.Game.Rulesets.Mania.Tests/TestMathUtilitiesDownHeap.cs

- A screenshot of the old coverage results:


75% TrianglesV2
40% TrianglesDrawNode
0% getClampedQuad(Axes,Vector2,Vector2):Quad
0% Draw(IRenderer):void
100% Source:TrianglesV2

- A screenshot of the new coverage results:



TrianglesDrawNode	41%	47/80
MarkBranchCovered(int)	100%	0/6
getClampedQuad(Axes,Vector2,Vector2)	100%	0/17
PrintCoverage()	100%	0/9

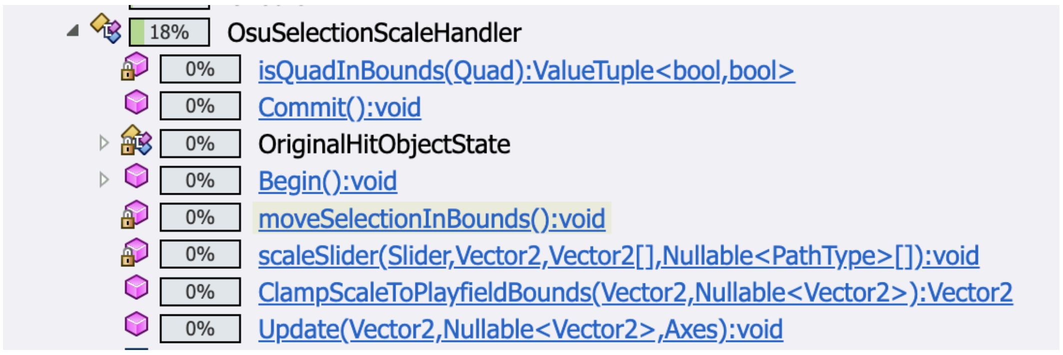
- Each conditional branch (if and else statements) within the downHeap function is covered by specific test cases. Test cases like DownHeap_FirstChildComparisonTrue, DownHeap_FirstChildComparisonFalse, DownHeap_BreakCondition, and DownHeap_NoBreakCondition ensure all possible outcomes of conditional branches are tested. The tests validate the function's behaviour comprehensively, ensuring it correctly handles heap operations (such as comparisons and assignments) across different input arrays (keys) and comparator conditions (comparer).

Group member 2 : *Lama Abboud:*

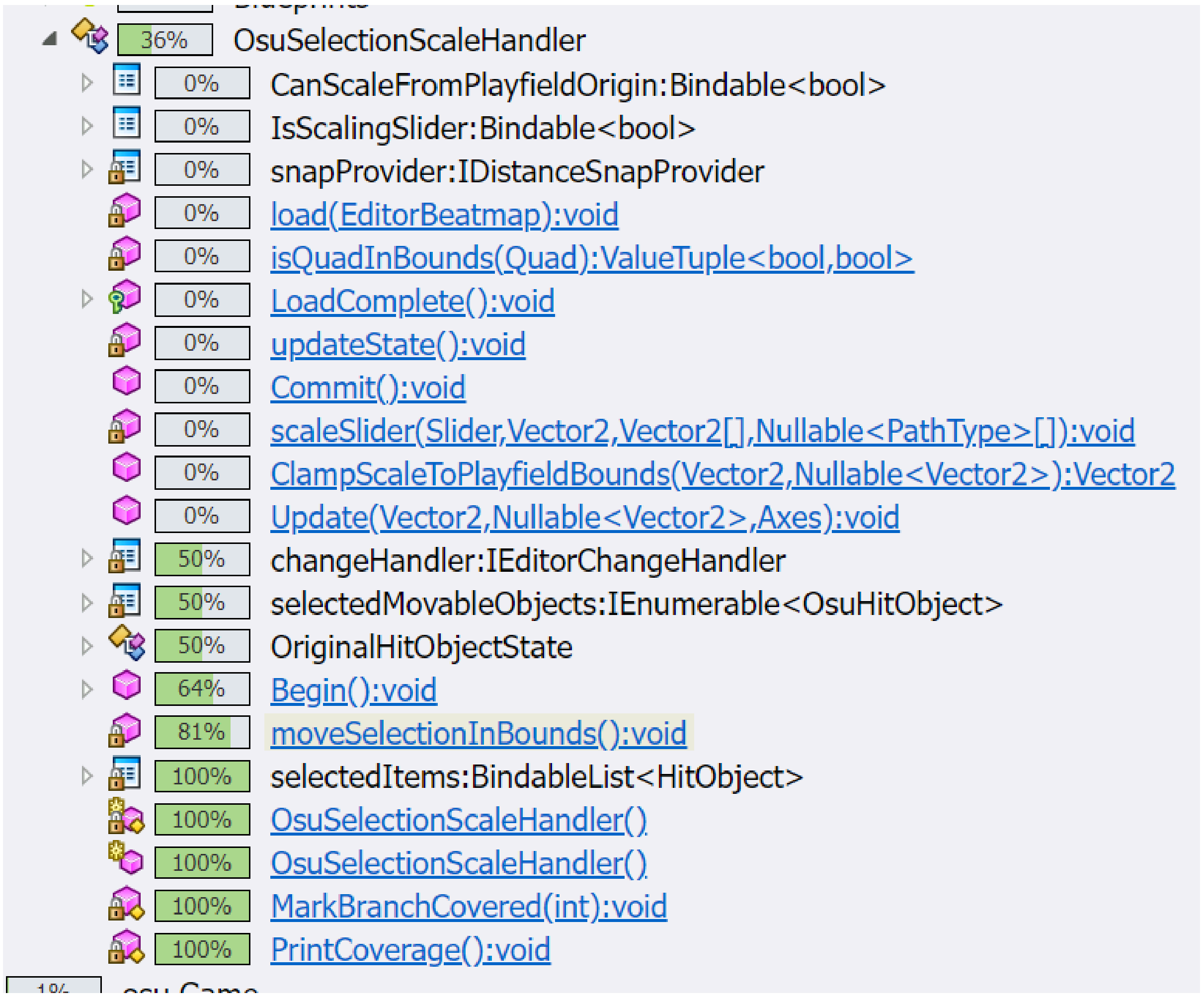
Test 1: OsuSelectionScaleHandlerTests

- A link to a commit on the forked repository of the new test: https://github.com/marwat16/SEP_group122_osu/blob/master/osu.Game.Rulesets.Osu.Tests/OsuSelectionScaleHandlerTests.cs

- A screenshot of the old coverage results:


18% OsuSelectionScaleHandler
0% isQuadInBounds(Quad):ValueTuple<bool,bool>
0% Commit():void
0% OriginalHitObjectState
0% Begin():void
0% moveSelectionInBounds():void
0% scaleSlider(Slider,Vector2,Vector2[],Nullable<PathType>[]):void
0% ClampScaleToPlayfieldBounds(Vector2,Nullable<Vector2>):Vector2
0% Update(Vector2,Nullable<Vector2>,Axes):void

- A screenshot of the new coverage results:



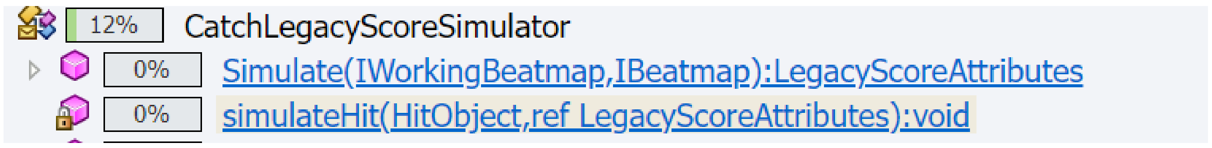
36%	OsuSelectionScaleHandler
0%	CanScaleFromPlayfieldOrigin:Bindable<bool>
0%	IsScalingSlider:Bindable<bool>
0%	snapProvider:IDistanceSnapProvider
0%	load(EditorBeatmap):void
0%	isQuadInBounds(Quad):ValueTuple<bool,bool>
0%	LoadComplete():void
0%	updateState():void
0%	Commit():void
0%	scaleSlider(Slider,Vector2,Vector2[],Nullable<PathType>[]):void
0%	ClampScaleToPlayfieldBounds(Vector2,Nullable<Vector2>):Vector2
0%	Update(Vector2,Nullable<Vector2>,Axes):void
50%	changeHandler:IEditorChangeHandler
50%	selectedMovableObjects:IEnumerable<OsuHitObject>
50%	OriginalHitObjectState
64%	Begin():void
81%	moveSelectionInBounds():void
100%	selectedItems:BindableList<HitObject>
100%	OsuSelectionScaleHandler()
100%	OsuSelectionScaleHandler()
100%	MarkBranchCovered(int):void
100%	PrintCoverage():void

- The initial branch coverage of the `OsuSelectionScaleHandler` class was 18%. By enhancing the `MoveSelectionInBounds` method with comprehensive tests, the coverage of the MoveSelectionInBound` method increased from 0% to 81%, and the overall coverage of the `OsuSelectionScaleHandler` class increased to 36%.

Test 2: simulateHit.tester.cs

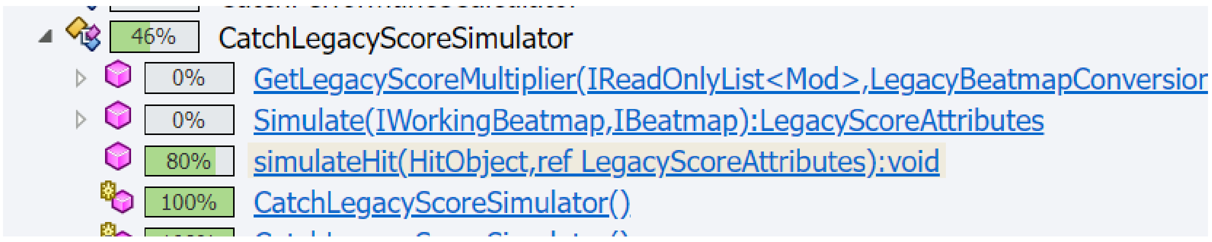
- A link to a commit on the forked repository of the new test: https://github.com/marwat16/SEP_group122_osu/blob/master/osu.Game.Rulesets.Catch.Tests/simulateHit.tester.cs

- A screenshot of the old coverage results:



12%	CatchLegacyScoreSimulator
0%	Simulate(IWorkingBeatmap,IBeatmap):LegacyScoreAttributes
0%	simulateHit(HitObject,ref LegacyScoreAttributes):void

- A screenshot of the new coverage results:



```
◢ 46%   CatchLegacyScoreSimulator
  ▷ 0%    GetLegacyScoreMultiplier(IReadOnlyList<Mod>,LegacyBeatmapConversior
  ▷ 0%    Simulate(IWorkingBeatmap,IBeatmap):LegacyScoreAttributes
    80%   simulateHit(HitObject,ref LegacyScoreAttributes):void
    100%  CatchLegacyScoreSimulator()
```

- The initial branch coverage of the `CatchLegacyScoreSimulator` class was 12%. By enhancing the `simulateHit` method with comprehensive tests, the coverage of the `simulateHit` method increased from 0% to 80%, and the overall coverage of the `CatchLegacyScoreSimulator` class increased to 46%.

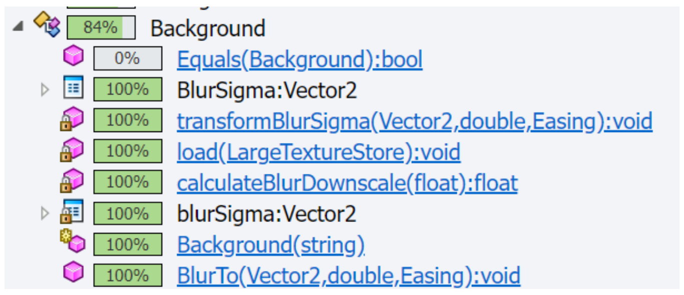
Group member 3 : *Daphne Lefevre*

Test 1: TestSceneBackgroundEquals

- A link to a commit on the forked repository of the new test:

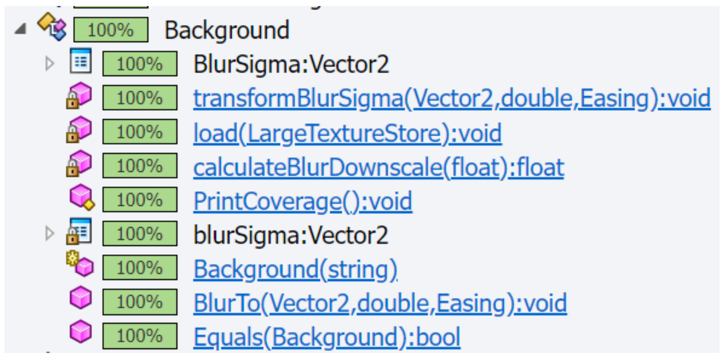
https://github.com/marwat16/SEP_group122_osu/blob/Daphn%C3%A9/osu.Game.Tests/Visual/Background/TestSceneBackgroundEquals.cs

- A screenshot of the old coverage results:



```
◢ 84%   Background
    0%    Equals(Background):bool
  ▷ 100%  BlurSigma:Vector2
    100%  transformBlurSigma(Vector2,double,Easing):void
    100%  load(LargeTextureStore):void
    100%  calculateBlurDownscale(float):float
  ▷ 100%  blurSigma:Vector2
    100%  Background(string)
    100%  BlurTo(Vector2,double,Easing):void
```

- A screenshot of the new coverage results:



```
◢ 100%  Background
  ▷ 100%  BlurSigma:Vector2
    100%  transformBlurSigma(Vector2,double,Easing):void
    100%  load(LargeTextureStore):void
    100%  calculateBlurDownscale(float):float
    100%  PrintCoverage():void
  ▷ 100%  blurSigma:Vector2
    100%  Background(string)
    100%  BlurTo(Vector2,double,Easing):void
    100%  Equals(Background):bool
```

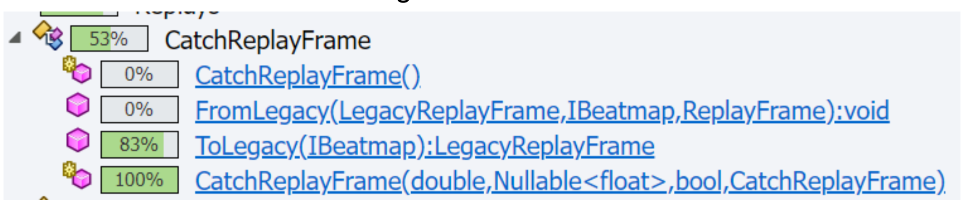
- The initial branch coverage of the `Background` class was 84%. By enhancing the `Equals` method with comprehensive tests, the coverage of the `Equals` method increased from 0% to 100%, and the overall coverage of the `Background` class increased to 100%. This was achieved by testing all possible conditions: comparison with `null`, the same instance, different types, and different textures. These tests ensured that all branches were covered.

Test 2: TestSceneCatchReplayFrameFromLegacy

- A link to a commit on the forked repository of the new test:

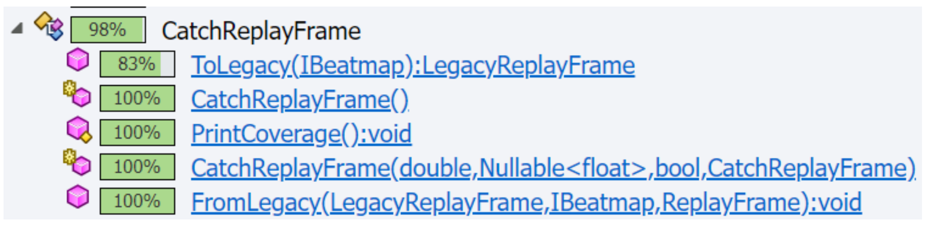
https://github.com/marwat16/SEP_group122_osu/blob/Daphn%C3%A9/osu.Game.Rulesets.Catch.Tests/TestSceneCatchReplayFrameFromLegacy.cs

- A screenshot of the old coverage results:



53%	CatchReplayFrame
0%	CatchReplayFrame()
0%	FromLegacy(LegacyReplayFrame,IBeatmap,ReplayFrame):void
83%	ToLegacy(IBeatmap):LegacyReplayFrame
100%	CatchReplayFrame(double,Nullable<float>,bool,CatchReplayFrame)

- A screenshot of the new coverage results:



98%	CatchReplayFrame
83%	ToLegacy(IBeatmap):LegacyReplayFrame
100%	CatchReplayFrame()
100%	PrintCoverage():void
100%	CatchReplayFrame(double,Nullable<float>,bool,CatchReplayFrame)
100%	FromLegacy(LegacyReplayFrame,IBeatmap,ReplayFrame):void

- The initial branch coverage of the `CatchReplayFrame` class was 53%, with the `FromLegacy` method having 0% coverage. After adding comprehensive tests and branch instrumentation for the `FromLegacy` method, the coverage of `CatchReplayFrame` increased to 98%, with `FromLegacy` achieving 100% coverage. This was accomplished by thoroughly testing all possible branches within the method, including different states of `Dashing`, movement directions, and position comparisons. These tests ensured that all conditional paths were exercised, significantly enhancing the robustness and reliability of the `CatchReplayFrame` class.

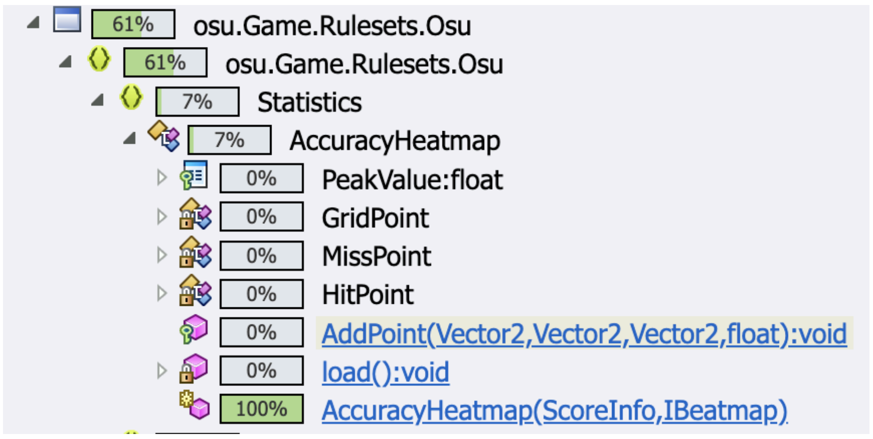
Group member 4 : *Marwa Tinmiouli*

Test 1: AddPointTests

- A link to a commit on the forked repository of the new test:

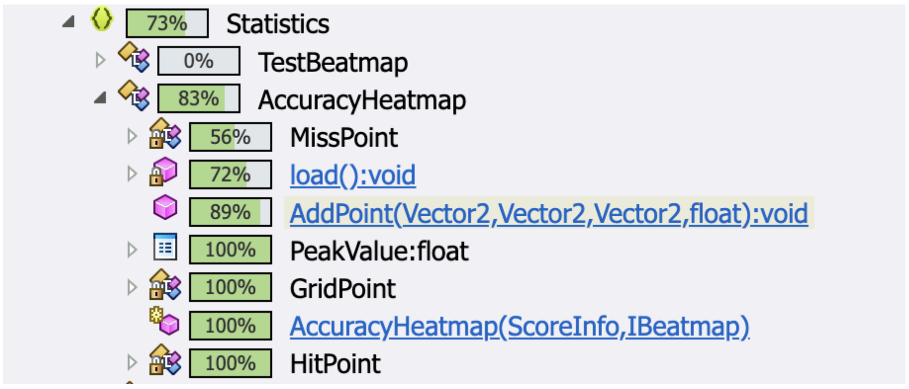
https://github.com/marwat16/SEP_group122_osu/blob/master/osu.Game.Rulesets.Osu/Statistics/AddPointTests.cs

- A screenshot of the old coverage results:



```
61%   osu.Game.Rulesets.Osu
  61%   osu.Game.Rulesets.Osu
    7%   Statistics
      7%   AccuracyHeatmap
        0%   PeakValue:float
        0%   GridPoint
        0%   MissPoint
        0%   HitPoint
        0%   AddPoint(Vector2,Vector2,Vector2,float):void
        0%   load():void
        100%   AccuracyHeatmap(ScoreInfo,IBeatmap)
```

- A screenshot of the new coverage results:



```
73%   Statistics
  0%   TestBeatmap
  83%   AccuracyHeatmap
    56%   MissPoint
    72%   load():void
    89%   AddPoint(Vector2,Vector2,Vector2,float):void
    100%   PeakValue:float
    100%   GridPoint
    100%   AccuracyHeatmap(ScoreInfo,IBeatmap)
    100%   HitPoint
```

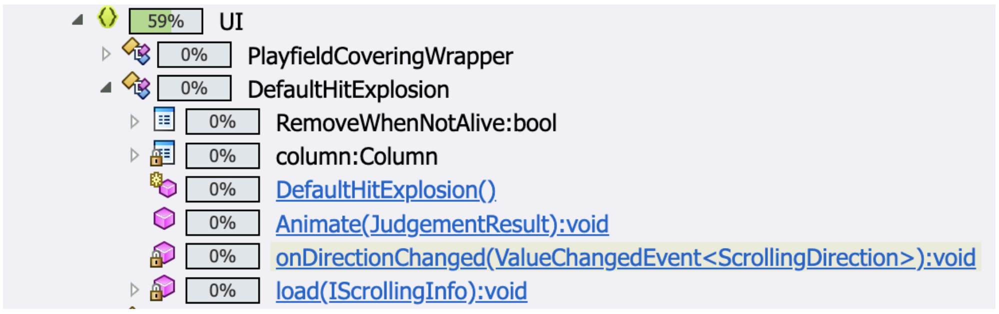
- The initial branch coverage of the `AccuracyHeatmap` class was 7%. By enhancing the `AddPoint` method with comprehensive tests, the coverage of the ‘AddPoint` method increased from 0% to 89%, and the overall coverage of the `AccuracyHeatmap` class increased to 83%.

Test 2: OnChangeDirectionTests

- A link to a commit on the forked repository of the new test:

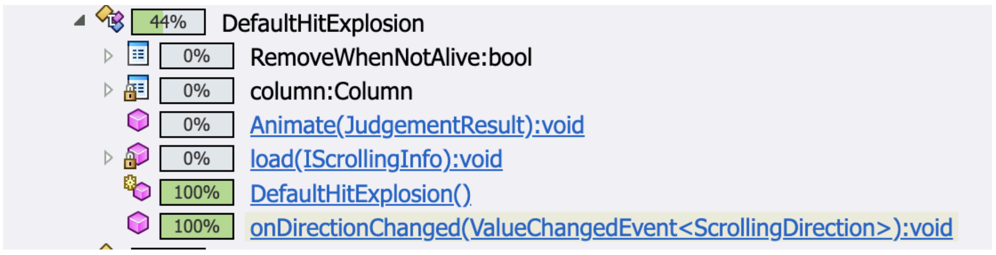
https://github.com/marwat16/SEP_group122_osu/blob/master/osu.Game.Rulesets.Mania/UI/OnChangeDirectionTests.cs

- A screenshot of the old coverage results:



```
59% UI
  0% PlayfieldCoveringWrapper
  0% DefaultHitExplosion
    0% RemoveWhenNotAlive:bool
    0% column:Column
    0% DefaultHitExplosion()
    0% Animate(JudgementResult):void
    0% onDirectionChanged(ValueChangedEvent<ScrollingDirection>):void
    0% load(IScrollingInfo):void
```

- A screenshot of the new coverage results:

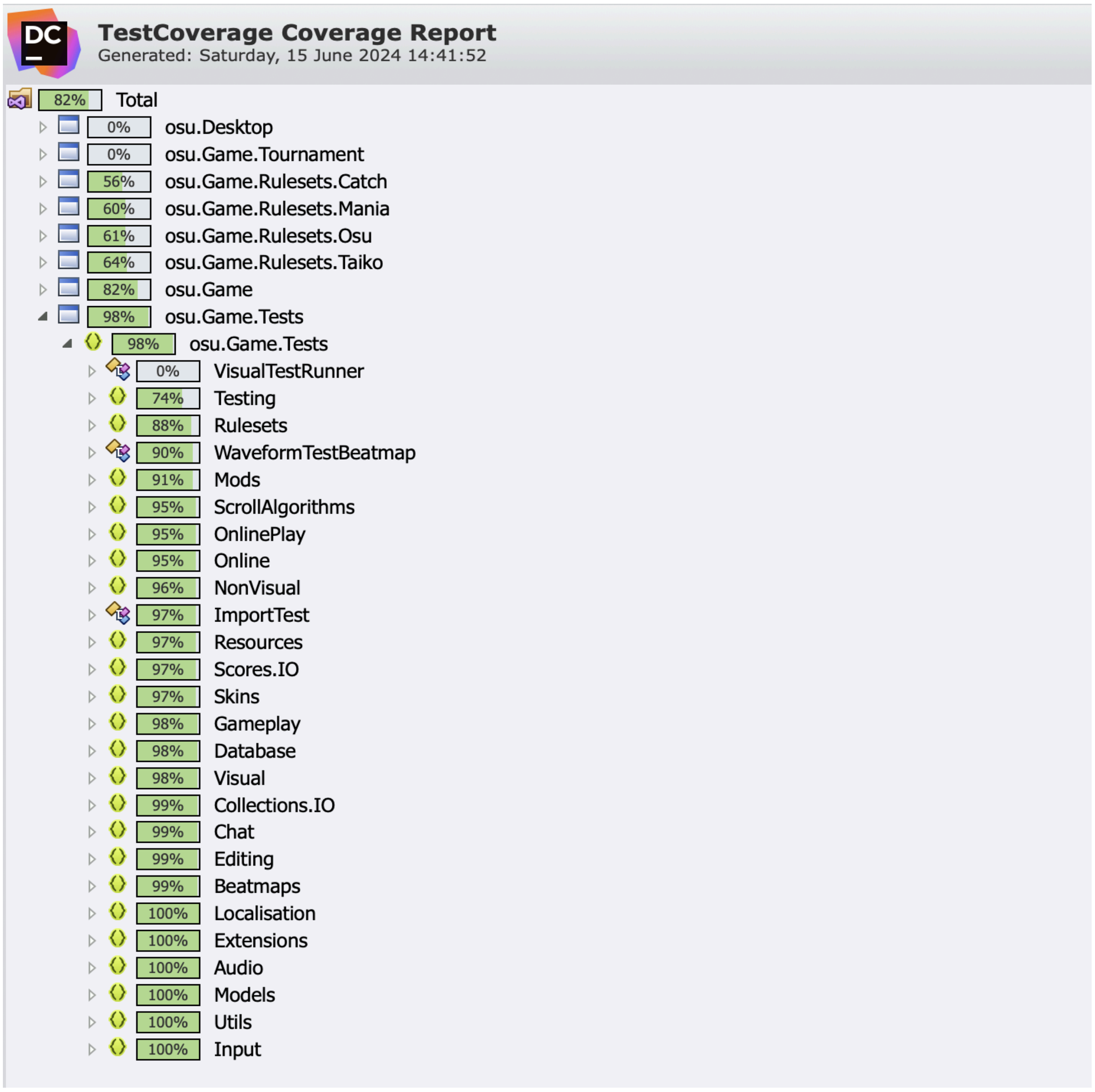


```
44% DefaultHitExplosion
  0% RemoveWhenNotAlive:bool
  0% column:Column
  0% Animate(JudgementResult):void
  0% load(IScrollingInfo):void
  100% DefaultHitExplosion()
  100% onDirectionChanged(ValueChangedEvent<ScrollingDirection>):void
```

- The initial branch coverage of the `DefaultHitExplosion` class was 0%. By enhancing the `OnDirectionChanged` method with comprehensive tests, the coverage of the ‘OnDirectionChanged` method increased from 0% to 100%, and the overall coverage of the `DefaultHitExplosion` class increased to 44%.

- **Overall**

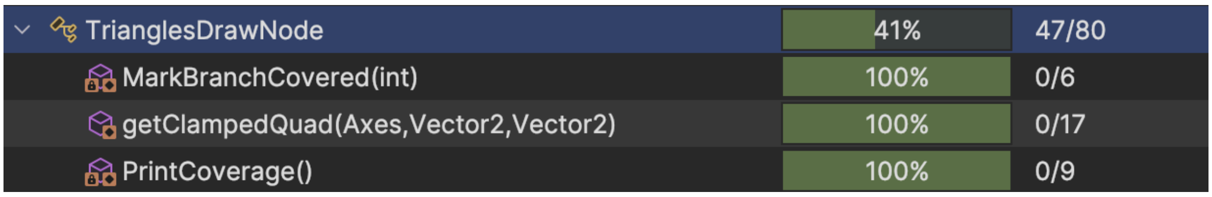
A screenshot of the old coverage results:



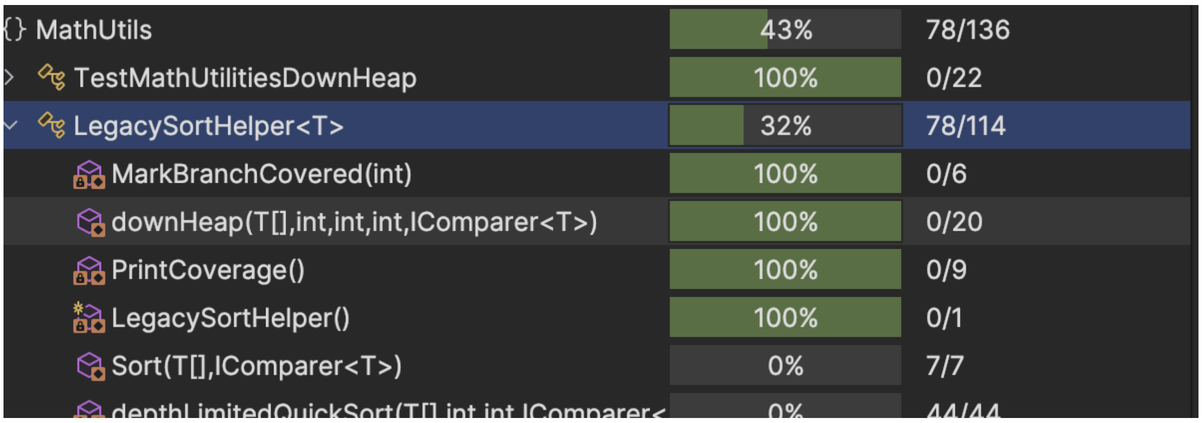
TestCoverage Coverage Report
Generated: Saturday, 15 June 2024 14:41:52

82%	Total
0%	osu.Desktop
0%	osu.Game.Tournament
56%	osu.Game.Rulesets.Catch
60%	osu.Game.Rulesets.Mania
61%	osu.Game.Rulesets.Osu
64%	osu.Game.Rulesets.Taiko
82%	osu.Game
98%	osu.Game.Tests
98%	osu.Game.Tests
0%	VisualTestRunner
74%	Testing
88%	Rulesets
90%	WaveformTestBeatmap
91%	Mods
95%	ScrollAlgorithms
95%	OnlinePlay
95%	Online
96%	NonVisual
97%	ImportTest
97%	Resources
97%	Scores.IO
97%	Skins
98%	Gameplay
98%	Database
98%	Visual
99%	Collections.IO
99%	Chat
99%	Editing
99%	Beatmaps
100%	Localisation
100%	Extensions
100%	Audio
100%	Models
100%	Utils
100%	Input

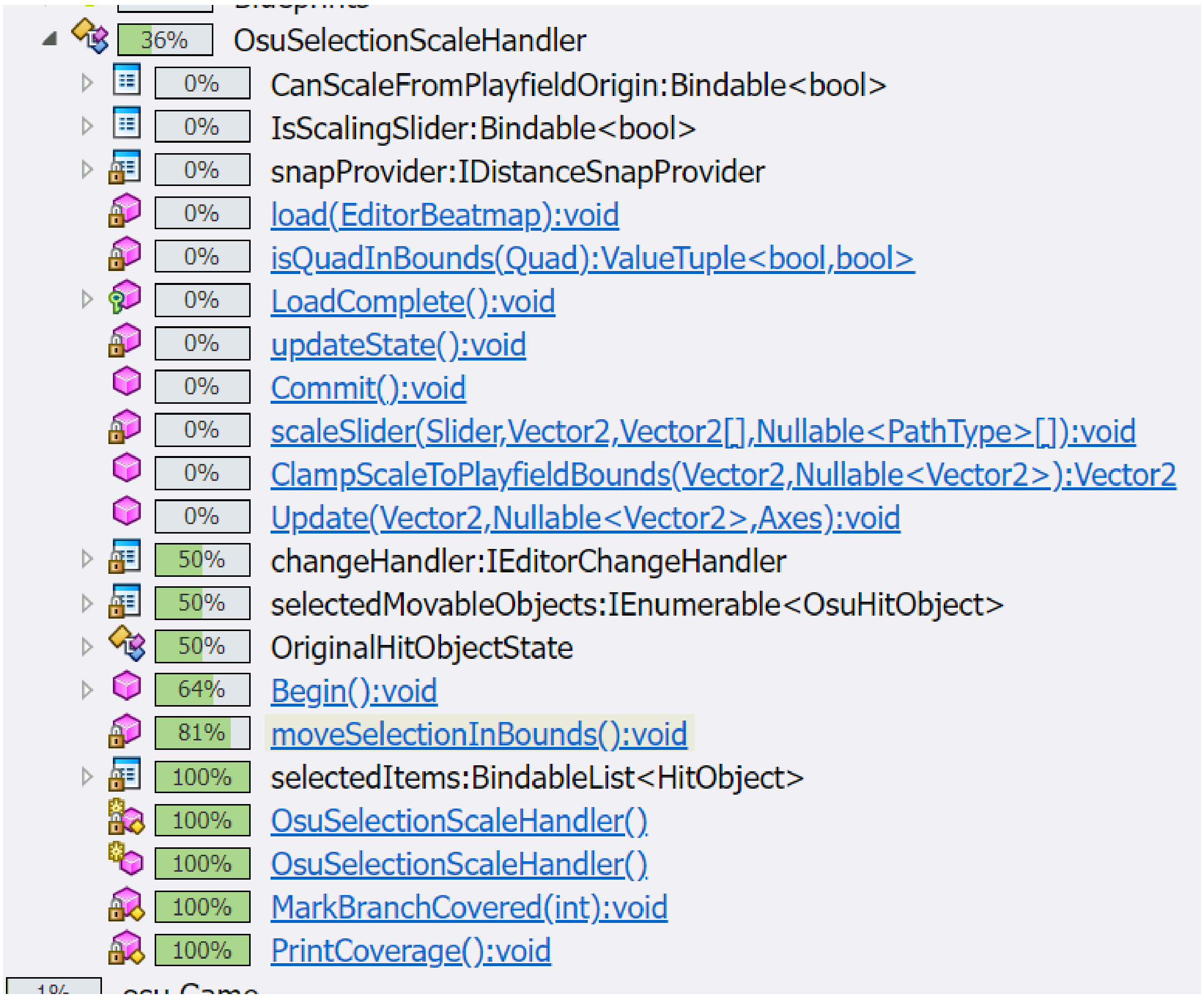
A screenshot of the new coverage results:



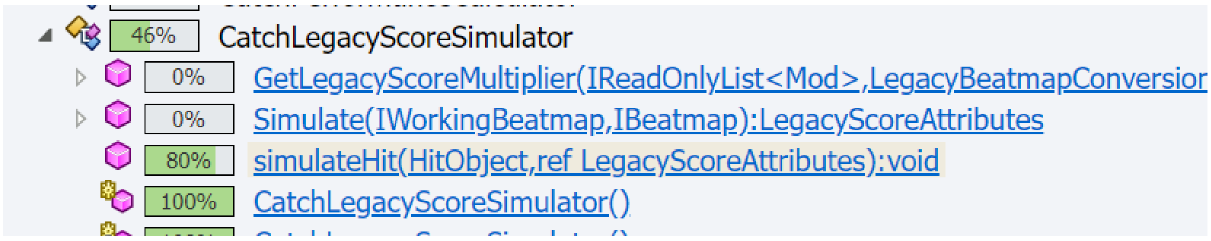
TrianglesDrawNode	41%	47/80
MarkBranchCovered(int)	100%	0/6
getClampedQuad(Axes,Vector2,Vector2)	100%	0/17
PrintCoverage()	100%	0/9



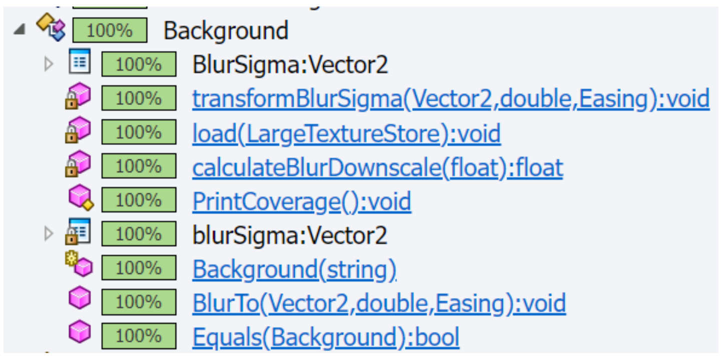
{} MathUtils	43%	78/136
TestMathUtilitiesDownHeap	100%	0/22
LegacySortHelper<T>	32%	78/114
MarkBranchCovered(int)	100%	0/6
downHeap(T[],int,int,int,IComparer<T>)	100%	0/20
PrintCoverage()	100%	0/9
LegacySortHelper()	100%	0/1
Sort(T[],IComparer<T>)	0%	7/7
depthLimitedQuickSort(T[],int,int,IComparer<	0%	44/44



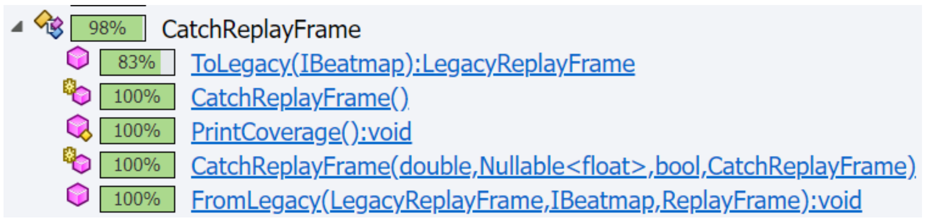
36%	OsuSelectionScaleHandler
0%	CanScaleFromPlayfieldOrigin:Bindable<bool>
0%	IsScalingSlider:Bindable<bool>
0%	snapProvider:IDistanceSnapProvider
0%	load(EditorBeatmap):void
0%	isQuadInBounds(Quad):ValueTuple<bool,bool>
0%	LoadComplete():void
0%	updateState():void
0%	Commit():void
0%	scaleSlider(Slider,Vector2,Vector2[],Nullable<PathType>[]):void
0%	ClampScaleToPlayfieldBounds(Vector2,Nullable<Vector2>):Vector2
0%	Update(Vector2,Nullable<Vector2>,Axes):void
50%	changeHandler:IEditorChangeHandler
50%	selectedMovableObjects:IEnumerable<OsuHitObject>
50%	OriginalHitObjectState
64%	Begin():void
81%	moveSelectionInBounds():void
100%	selectedItems:BindableList<HitObject>
100%	OsuSelectionScaleHandler()
100%	OsuSelectionScaleHandler()
100%	MarkBranchCovered(int):void
100%	PrintCoverage():void
1%	osu.Game



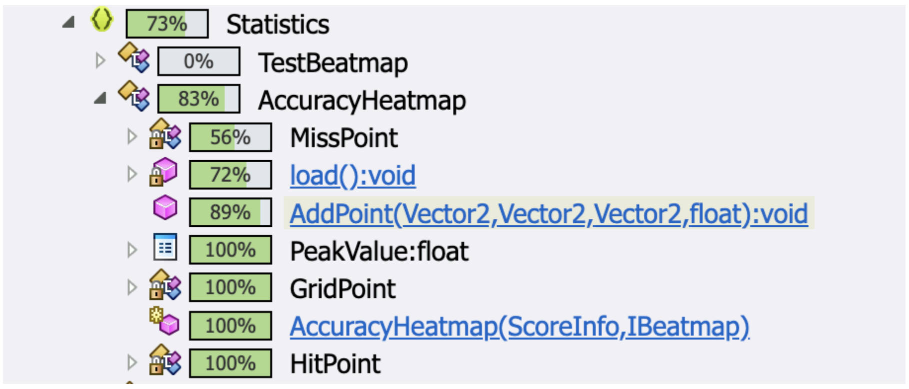
46%	CatchLegacyScoreSimulator
0%	GetLegacyScoreMultiplier(IReadOnlyList<Mod>,LegacyBeatmapConversior
0%	Simulate(IWorkingBeatmap,IBeatmap):LegacyScoreAttributes
80%	simulateHit(HitObject,ref LegacyScoreAttributes):void
100%	CatchLegacyScoreSimulator()



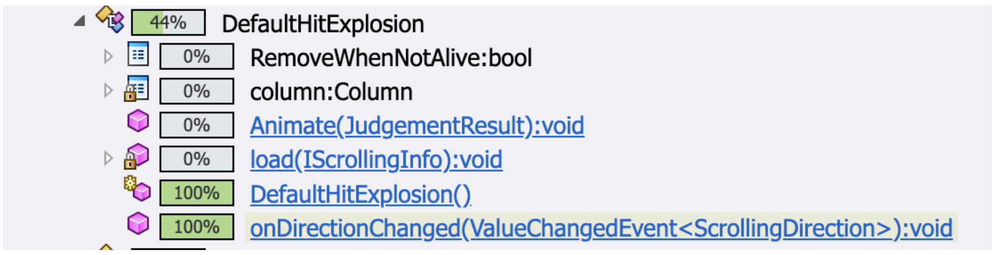
```
100% Background
  100% BlurSigma:Vector2
  100% transformBlurSigma(Vector2,double,Easing):void
  100% load(LargeTextureStore):void
  100% calculateBlurDownscale(float):float
  100% PrintCoverage():void
  100% blurSigma:Vector2
  100% Background(string)
  100% BlurTo(Vector2,double,Easing):void
  100% Equals(Background):bool
```



```
98% CatchReplayFrame
  83% ToLegacy(IBeatmap):LegacyReplayFrame
  100% CatchReplayFrame()
  100% PrintCoverage():void
  100% CatchReplayFrame(double,Nullable<float>,bool,CatchReplayFrame)
  100% FromLegacy(LegacyReplayFrame,IBeatmap,ReplayFrame):void
```



```
73% Statistics
  0% TestBeatmap
  83% AccuracyHeatmap
    56% MissPoint
    72% load():void
    89% AddPoint(Vector2,Vector2,Vector2,float):void
    100% PeakValue:float
    100% GridPoint
    100% AccuracyHeatmap(ScoreInfo,IBeatmap)
    100% HitPoint
```



```
44% DefaultHitExplosion
  0% RemoveWhenNotAlive:bool
  0% column:Column
  0% Animate(JudgementResult):void
  0% load(IScrollingInfo):void
  100% DefaultHitExplosion()
  100% onDirectionChanged(ValueChangedEvent<ScrollingDirection>):void
```

- ## Statement of individual contributions

Main tasks:

A. Finding the proper GitHub project
During several online meetings, we all came to osu -our final choice

B. Coverage measurement by an existing tool
Same as the previous task, we all struggled to understand the concepts of coverage and testing, in general, and how to run a measurement using dotCover

C. Choosing 2 functions and conduct the coverage improvement for each
Each one of us worked on this individual task, but we, of course, planned meetings in person to help each other when needed and put everything up on GitHub.