The phase II implementations involve programming to the API developed during the Phase I. The platform developed during Phase I is a tank battle game which involves a 2-d playfield of sizes varying from 5x5 to 50x20. Computer controlled players (tanks) can number between 2 and 10. The playfield can include a variety of obstacles with set characteristics.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Obstacle Name | Destructible | Traversable | Does Damage | Explodes w/splash damage | Shots pass through |
| Bush | Yes | Yes | No | No | No |
| Tree | Yes | Yes | No | No | No |
| Crate | Yes | Yes | No | Yes | No |
| Rock | Yes | No | Yes | No | No |
| Tank Buster | No | No | Yes | No | No |
| Water | No | No | Yes | No | Yes |

Tanks have a number of attributes which impact their ability to shoot, move, and explore the environment.

|  |  |
| --- | --- |
| Attribute Name | Description |
| Damage | Number of points of damage done by projectiles |
| Health | Number of points of damage the tank can absorb before being destroyed |
| Radar | Distance at which the tank can detect enemies and obstacles |
| AP | Action points available for each turn |
| Range | Distance projectiles move during a turn |
| Ammo | Number of shots available before requiring a reload |

At the beginning of a game, the tank program is given a number of SPECIAL points which can be allocated to increase its attributes. This can add considerable variability in the capabilities of a tank when strategically assigned.

There are three stages of solo tank testing with a number of levels in each stage. In all stages and levels, tanks will be allocated 2 SPECIAL points.

## Simple Navigation (30 points)

### Level 1 [For full points, tanks must ‘win’ 100% of these tests]

* No obstacles (other than the playfield boundaries)
* 1 or 2 immobile targets (Target)
* Tank has full visibility of the playfield

### Level 2 [Full points require 100% ‘win’ rate]

* No obstacles
* 1 or 2 Fleeing targets (Mouse)
* Tank has full visibility of the playfield

### Level 3 [Full points require 95% ‘win’ rate]

* No obstacles
* 1 immobile or 1 fleeing target
* Tank has limited visibility of the playfield

## Obstacle Avoidance (30 points)

### Level 1 [For full points, tanks must ‘win’ 100% of these tests]

* Sparse (non-damaging) obstacles
* 1 or 2 immobile targets
* Tank has full visibility of playfield

### Level 2 [For full points, tanks must ‘win’ 100% of these tests]

* Sparse (non-damaging) obstacles
* 1 or 2 fleeing targets
* Tank has full visibility of playfield

### Level 3 [For full points, tanks must ‘win’ 90% of these tests]

* Sparse (non-damaging) obstacles
* 1 immobile or 1 fleeing target
* Tank has limited visibility of playfield

## Offensive Opponent (30 points)

### Level 1 [For full points, tanks must ‘win’ 90% of these tests]

* No obstacles
* 1 or 2 Immobile targets that can shoot back
* Tank has full visibility of playfield

### Level 2 [For full points, tanks must ‘win’ 90% of these tests]

* Sparse obstacles
* 1 or 2 immobile targets that can shoot back
* Tank has full visibility of playfield

### Level 3 [For full points, tanks must ‘win’ 80% of these tests]

* Sparse obstacles
* 1 or 2 immobile targets that shoot back
* Tank has limited visibility

There are two additional stages of testing that will pit student tanks against other student tanks for bonus points.

## Head to Head (18 bonus points available)

### Level 1 [1 SPECIAL point]

* Tanks will be randomly matched against one other student tank
* Playfields will be random rectangles between 5 x 5 and 50 x 20
* No obstacles
* Visibility limited

### Level 2 [2 SPECIAL points]

* Tanks will be randomly matched against one other student tank
* Playfields will be random rectangles between 5 x 5 and 50 x 20
* Sparse obstacles (non-damaging)
* Visibility limited

### Level 3 [3 SPECIAL points]

* Randomly matched 1 v 1
* Random Sized playfields
* Medium obstacles (all types)
* Full field visibility

## Free For All (18 bonus points available)

### Level 1 [1 SPECIAL point]

* All student tanks in the same playfield. Playfields will range from 15 x 15 to 50 x 20
* No obstacles
* Limited Visibility

### Level 2 [2 SPECIAL points]

* All student tanks [15 x 15 to 50 x 20]
* Sparse obstacles (non-damaging)
* Limited Visibility

### Level 3 [4 SPECIAL points]

* All student tanks on same playfield [15 x 15 to 50 x 20]
* Medium obstacles (all types)
* Full visibility

# Scoring

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Simple Navigation | 100% | 90-99% | 80-89% | 70-79% | 60-69% | < 60% |
| Level 1 | 10 | 8 | 6 | 4 | 2 | 0 |
| Level 2 | 10 | 8 | 6 | 4 | 2 | 0 |
| Level 3 | 10 | 10 | 8 | 6 | 4 | 2 |
| Obstacle Avoidance | 100% | 90-99% | 80-89% | 70-79% | 60-69% | < 60% |
| Level 1 | 10 | 8 | 6 | 4 | 2 | 0 |
| Level 2 | 10 | 8 | 6 | 4 | 2 | 0 |
| Level 3 | 10 | 10 | 8 | 6 | 4 | 2 |
| Offensive Opponent | 100% | 90-99% | 80-89% | 70-79% | 60-69% | < 60% |
| Level 1 | 10 | 10 | 8 | 6 | 4 | 2 |
| Level 2 | 10 | 10 | 8 | 6 | 4 | 2 |
| Level 3 | 10 | 10 | 10 | 8 | 6 | 4 |

# Bonus

|  |  |  |  |
| --- | --- | --- | --- |
| Head to Head | 1st Place | 2nd Place | 3rd Place |
| Level 1 | 3 | 2 | 1 |
| Level 2 | 3 | 2 | 1 |
| Level 3 | 3 | 2 | 1 |
| Free For All | 1st Place | 2nd Place | 3rd Place |
| Level 1 | 3 | 2 | 1 |
| Level 2 | 3 | 2 | 1 |
| Level 3 | 3 | 2 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Level 1 | Level 2 | Level 3 | Total |
| Simple Navigation |  |  |  |  |
| Obstacle Avoidance |  |  |  |  |
| Offensive Opponent |  |  |  |  |
| Head To Head |  |  |  |  |
| Free For All |  |  |  |  |
| Totals |  |  |  |  |

# Maintenance Activities (93/100)

* Maintenance team didn’t produce a Tank for the game so the performance portion for Phase II will have to do with activities surrounding the upkeep and support of the platform chosen by the class for phase II. The above description of the tanks and their scoring is included for completeness.
* Team closed 40 of 43 issues.
* Completed 37 of 44 merge requests and closed 7. No outstanding merge requests.
* Started and monitored Discord channel for teams developing tanks. Responsiveness excellent.
* Instituted CI which included unit and feature tests. Pipelines were consistently run and passing.
* Continuous bug fixes and upgrades which was a two-edged sword since additional features in the API meant tank development went through a couple start-stop phases while teams tried to keep up with changes.
* One thing we should have developed and distributed would be a post-delivery plan. These plans typically include:
  + Details about who will perform maintenance (we had defined that with a separate team)
  + A formal description of supporting procedures (e.g. process for submitting bug reports, RFEs, RFCs, etc.)
  + What resources will be set aside for maintenance (not that important in our case but important when undertaken at the institutional level)
  + Outline for quality assurance
  + Existing and additional testing (i.e. regression tests created from bug reports)
  + Description of the branching, forking, and merging policies to be used.
  + A complete and formal description of the system when it went into maintenance mode.
  + Details about how long and under what conditions the system will be maintained.
* Since this wasn’t part of the initial requirements for the maintenance team it’s not part of the grading for them.

# Code Review (90/100)

* Lots of repo activity between the selection of the platform and the final release.
* Fairly well distributed activity save for a couple maintenance team members which did not have significant input to the support of the platform.
* The src/ directory contains a mixture of example tank AIs and platform functional code. These should be separated since there’s a bright line between the two. Examples should occupy testing and/or documentation and API sub-directories. Keep in mind we would like others to pick this up and run with it in the future. Clear organization will help future teams an easier time understanding the structure.
* A minor nit would be to separate include files from source files. Due to the nature of the platform, parts of the interface (i.e. include files) are delivered but the associated source files are not. For clarity and organization sake the files delivered with the API should occupy a different directory.
* The documentation on the wiki page is outdated and contains much of the information from the original Slacker’s repo. In as far as is possible, this should be updated.
* Some of the #ifdef guard labels violate C++ naming convention. Identifiers that start with an underscore are generally considered to be reserved and \*any\* identifier with double underscore is reserved.
* What’s the buildsrc/ directory for?
* Some of your draw routines contain direct calls to openGL routines. This tightly couples the platform to a particular graphics package and will make it harder to change renderers or port to other platforms. There should be an abstraction layer that the platform code calls and then calls on to the architecture specific package.
* Several files have significant commented out code (e.g. game.cpp)
* There are multiple GameField constructors which could be consolidated by using default and optional parameters. The multiple constructors violate DRY and DIE principles.
* Lots of DRY and DIE in the GameField file also commented out code.
* Function / Method headers need more description.
* GameField::nextTurn needs refactoring. Hard to read, too large, does too much
* The ::getWinner method seems to pick the last tank with health as the winner without taking into account other tanks with (perhaps) more health, higher percentage of health remaining, number of hits / shots, etc.
* Why are there globals being used (global.h)? This is lazy, causes bugs that can be hard to track down, prevents the code from being parallelized, and causes kittens to be killed every time it’s compiled.
* Projectile constructor does too much work and has a pile of magic numbers.
* Nice design where projectileActor implements Actor and behaves like a tank.
* Somewhere in the documentation make sure you mention the use of the third party SOIL.h file.
* The DRY and DIE violations in TankDrawable . . .
* In general the platform isn’t in bad shape. I think another team could pick it up and make some good progress in a short amount of time. I’m seriously considering using it again in a future Soft Eng class.