Viikko 4.

6.2. Perjantai – 9.2. Maanantai

Aloitin viikkoni refaktoroimalla koko piirto-operaation omaan sisäiseen luokkaansa, jolloin sen käyttämisestä tuli erittäin helppoa ja GUI:n tehtävät vähenivät ja siitäkin tuli paljon selkeämpi mielestäni. Sain myös toteutettua dynaamisuutta tämän uuden luokan avulla A*:ni ja nyt A*:n toiminta tapahtui niin, että samalla kun algoritmia ajetaan, niin se piirtää itseään karttaan. Tällöin karttaan tulee lopussa reitti ja polku A → B.

Dynaamisuudelle oli kuitenkin erityinen ja tärkeä syy; halusin, että piirrosta tulee reaaliaikainen, jotta ei tarvitsisi aina odottaa A* suoritusta ja erikseen piirtää karttaa. Tämän reaaliaikaisuuden avulla pystyn toteuttamaan projektini viimeisen vaiheen: liikkuvan maalin. Liikkuva maali tässä tapauksessa tarkoittaa hiiren klikkausta, joka tehdään kesken A* algoritmin suoritusta.

Tutustuin myös viikonlopun aikana adaptiiviseen A*-algoritmiin tai tarkemmin algoritmiin nimeltä MT-AA* eli Moving-Target Adaptive A*. Tämä algoritmi on laajennos A* algoritmille ja sen toteuttaminen ei ole kovin vaikeaa. Kuitenkaan materiaalia algoritmeista, jotka soveltuvat liikkuviin maaleihin ei ole kovin helppo löytää, vaan suurin osa niistä on tieteellisiä julkaisuja omista algoritmeista kuten D*, D*-lite, MTS, RTA* jne. Tutustuin moneen näistä algoritmiin viikonlopun aikana ja täytyy myöntää ettei pseudo-koodin lukeminen ole aina ihan helppoa.

10.2. Tiistai – 11.2. Keskiviikko

Jatkoin ideaani liikkuvasta maalista ja tulin siihen tulokseen, että minun on aluksi luotava jokin hidas ratkaisu liikkuvalle maalille. Ratkaisu olikin yllättävän yksinkertainen ja se löytyi jopa mainintana useassa tieteellisessä julkaisussa reitinhakualgoritmeista, joissa on liikkuva maali. Ratkaisu jonka toteutin, on jopa käytössä tietyissä tietokonepeleissä, joissa kontrolloitavia hahmoja ei ole useita, mutta käyttöä A* on. Ratkaisu hitaalle menetelmälle oli siis tehdä uusi A* haku, niin että OpenSet ja ClosedSet tyhjennettiin ja aloitus- ja lopetuskoordinaatit muutettiin vastaamaan hiiren klikkausta. Tämä oli yllättävän tehokas, koska algoritmini heurestiset arvot lasketaan lennosta, joten heurestisia arvoja ei tarvitse käydä muuttamassa erikseen.

Nyt kun ohjelmani suoritti ja ajoi A*, niin klikkaamalla kesken hakua muutettiin maalin paikkaa, jonka A* havaitsi ja osasi suorittaa haun ”uudestaan”. Todellisuudessa hakua ei aloitettu uudestaan, vaan tein asian hieman tehokkaammalla tavalla tyhjentämällä listat.

12.2. Torstai

Olen nyt varma, että työni loppupaino tulee olemaan mainitussa MT-AA* haussa, joka ei suorita hakua uudestaan, vaan muistaa edellisten hakujen perusteella, että mikä on kaikista paras solmu jatkaa hakua ja minne suuntaan hakua kannattaisi jatkaa. On kuitenkin jännä huomata tässä vaiheessa projektia, että algoritmin on aikavaativuudeltaan O(|E| * |V|), jossa E tarkoittaa kaarien määrää ja V solmujen. Todellisuudessa vakiokertoimet kuitenkin A* ovat huomattavasti alhaisemmat, kuin Dijkstrassa ja MT-AA* on taas vakiokertoimiltaan A* nopeampi, mutta vai peräkkäisissä hauissa.

4. Viikko kokonaisuudessaan

Sain mielestäni paljon aikaan viikon aikana, vaikka hieman ongelmia oli/on vielä tieteellisten julkaisujen ymmärtämisessä ja pseudo-koodin lukemisessa. MT-AA* kuitenkin vaikuttaa parhaalta ratkaisulta ongelmaani, eikä sen toteuttaminen näytä olevan liian monimutkaista, vaikka aikaa ei ole

kauheasti enää jäljellä.