2019

# Technisch rapport


Raoul Wernert

1676356

3/9/2019

# Inleiding

In dit rapport staan de technische specificaties beschreven van het spel Multi-Memory. Multi-Memory is afgeleid van een bestaande memory opdracht voor het vak Educative games programmeren 1 van de minor Vernieuwend leren met ICT & games.

In dit document worden de complexe functionaliteiten besproken zoals de architectuur, hoe de game ineklaar zit door middel van een klassendiagram.

# Versiebeheer

| Versie | Datum | Aanpassingen |
|---|---|---|
| **1.0.0** | 03-02-2019 | Inleiding, Architectuur, Klassendiagram |
| **2.0.0** | 09-03-2019 | Voorblad, Uitbreiding inleiding, Versiebeheer, Inhoudsopgave, Uitbreiding klassendiagram, Functionaliteiten |

# Contents

# Architectuur

Uit de opdracht is gebleken dat het spel in een browser moet draaien. Vandaar dat ik gekozen heb om het spel in html/js/css te maken. Dit leek mij het meest doelgerichte zonder dat de speler allerei benodigdheden moest downloaden en/of installeren.

## Github pages

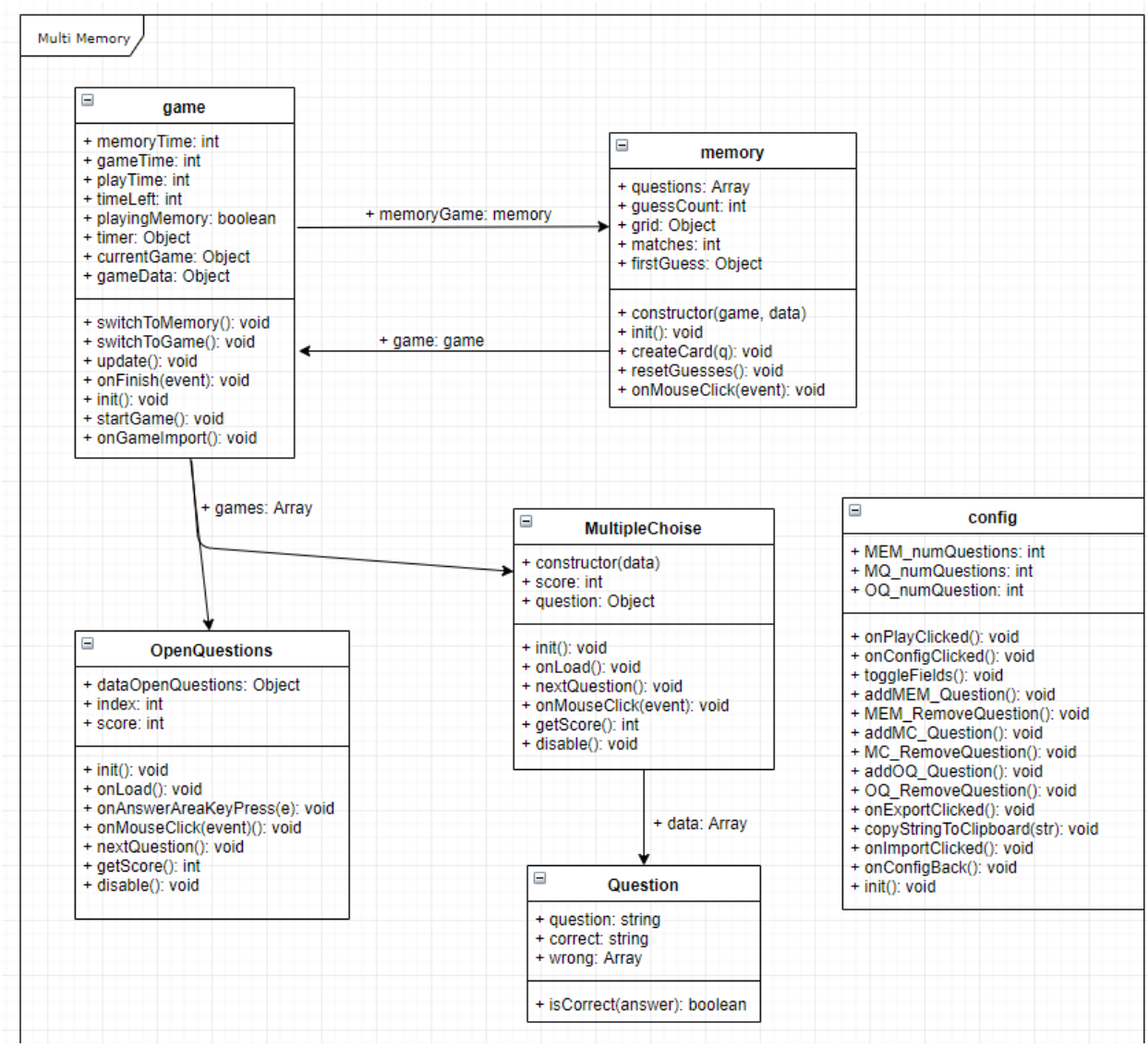
Om het daadwerkelijk laten draaien van het spel nog makkelijker te maken voor de gebruiker heb ik er voor gekozen om gebruik te maken van Github pages. Hierbij wordt de inhoud van een Github repository gehost door Github en is deze te bereiken via https://raoulwernert.github.io/OA-MINLEARB-14-GAME/.

## Wordpress

Helaas is het niet gelukt om de game volledig in Wordpress te hosten aangezien de game bestaat uit meerdere html/js/css files die niet overgezet kunnen worden. Ook is het niet mogelijk om het spel te imbedden via een object of iframe aangezien de pagina is opgedeelt in columnen en deze een breedte hebben dat kleinder is dat wat de game nodig zou hebben. Ik heb nu voor de oplossing gekozen om een knop in de Wordpress pagina te zetten die naar het spel linkedm waarop het op die manier wel te spelen is. Wordpress link: https://edhugames.hu.nl/raoul-wernert/oa-minlearb-14-game_raoulwernert_1676356-2/

## Klassendiagram

Dit is het klassendragram van de game. Ik heb er voor gekozen een klassediagram te gebruiken omdat dit een direcht overzicht geeft van hoe de game inelkaar zit.



# Functionaliteiten

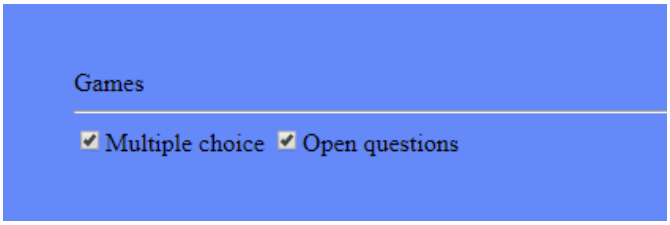
In dit gedeelte worden de meest complexe functionaliteiten beschreven aan de hand van diepergaande uitleg en snippits van stukjes code.

## Editor

In de game is het mogelijk om per minigame zelf de vragen en antwoorden in te voeren, te updaten en te verwijderen. De code snippets zijn terug te vinden in de klasse js/config.js.

### Games

Bij het games gedeelte kan er aangegeven worden welke minigames er mogelijk speelbaar zijn.

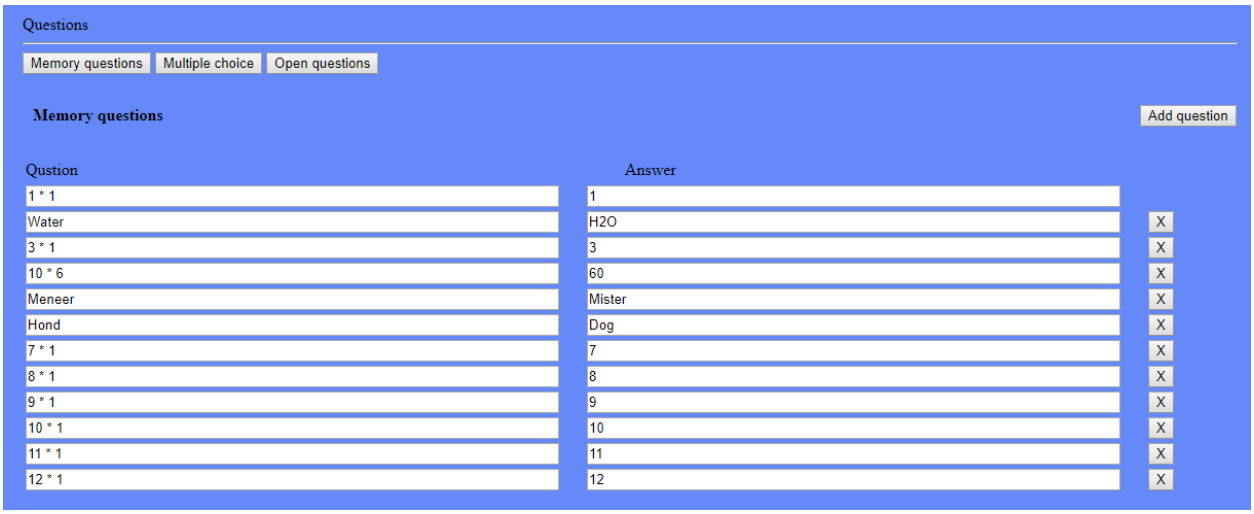



Natuurlijk is kan het spel alleen gespeeld worden als er ten minste één minigame aan staat. Hier zit er dan ook een beveiliging op dat er minimaal één minigame aan moet staan.

```
86      let mcChecked = $('#mc-checkbox').is(":checked");
87      let oqChecked = $('#oq-checkbox').is(":checked");
88      if(!mcChecked && !oqChecked){
89          alert('Please check at least 1 checkbox');
90          return;
91      }
```

## Questions

Het questions gedeelte bied de mogelijkheid om voor de memory game en voor de minigames de vragen en antwoorden in te voeren.




Hierboven zijn de vragen en antwoorden van de memory game weergegeven. Hierbij is het mogelijk om meer meer vraag en antwoorden toe te voegen door rechtsboven op de Add question knop te klikken. Ook is het mogelijk om vraag en antwoorden te verweideren, dit doe je door op de X knop te drukken aan de rechterkant van de vraag. Elke vraag en antwoord kan tussendoor aangepast worden zonder dat deze eerst verwijderd moet worden.

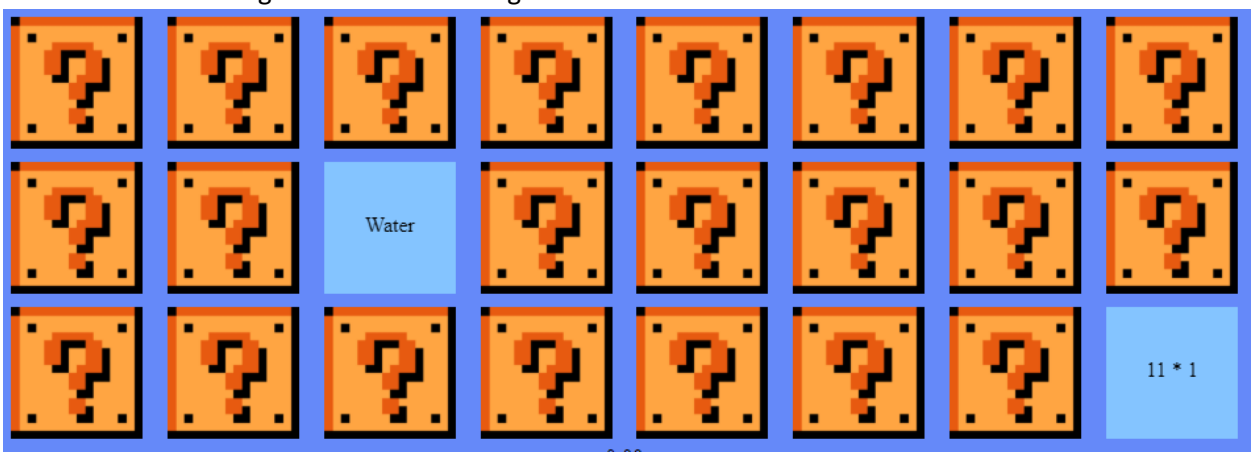
Hierbij de functionaliteit dat wordt uitgevoerd wanneer er op de Add question geklikt is:

```
20  const addMEM_Question = function () {
21      MEM_numQuestions++;
22      let tr = '<tr id=MEM-question'+MEM_numQuestions+'></tr>';
23      let question = '<td><input style="width: 95%" id="MEM-q-'+MEM_numQuestions+'"></td>';
24      let answer = '<td><input style="width: 95%" id="MEM-a-'+MEM_numQuestions+'"></td>';
25
26      tr = $(tr).append(question, answer);
27      if(MEM_numQuestions > 1){
28          let removeButton = '<td><button onclick="MEM_RemoveQuestion('+MEM_numQuestions+')">X</button></td>';
29          tr = $(tr).append(removeButton);
30      }
31      $('#MEM-table').append(tr);
32  }
```

Hierbij wordt gebruik gemaakt van de Jquery library (terug te vinden in js/jquery-3.3.1.js). Ik maak gebruik van Jquery omdat deze library veel regelt in weinig regels code. Bijvoorbeeld: om een element te selecteren in javascript moet je document.getElementById(‘#test’); doen, met Jquery kan je een element selecteren door alleen $(‘#test’); te doen waardoor de code een stuk leesbaarder blijft.

Omdat er minimaal één vraag en antwoord nodig is om het spel te kunnen spelen wordt er gekeken of dit de eerste vraag is, zo niet, dan wordt er een verwijder knop aan toegevoegd.

Voorbeeld van de vragen van hierboven ingame:




## Import/Export

Wanneer alle vragen en antwoorden zijn opgegeven is het natuurlijk niet optimaals wanneer deze vragen en antwoorden voor elk spel opnieuw ingevoerd moeten worden. Daarom heeft de editor export en inport functionaliteit.

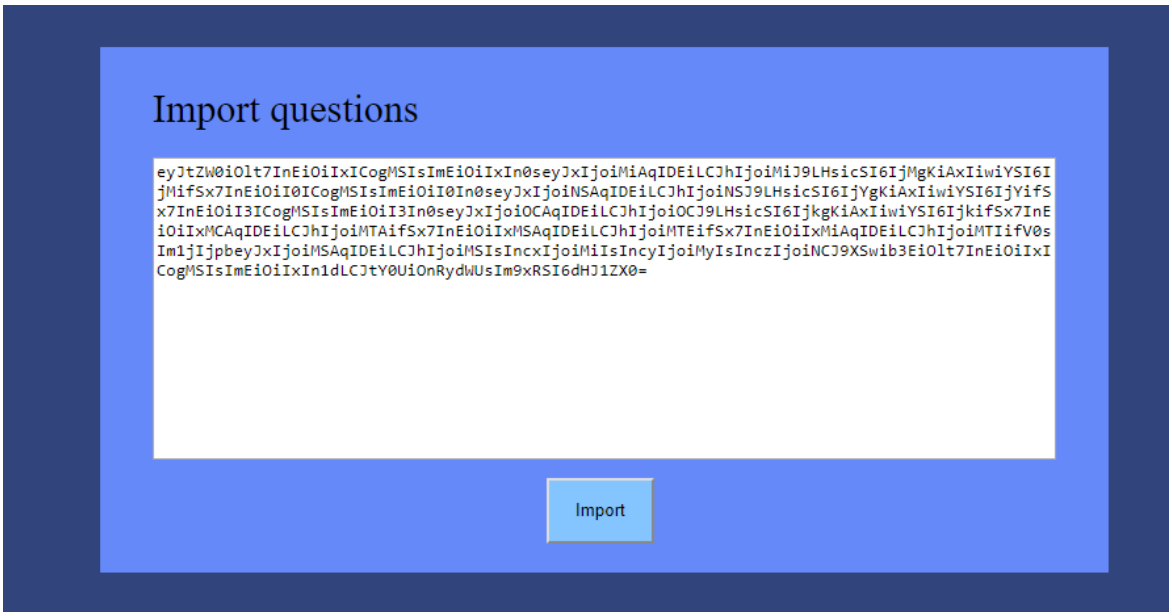
Hier een voorbeeld hoe de vragen en antwoorden voor de memory game worden geexporteerd.

```
93      for(let i = 1; i <= MEM_numQuestions; i++){
94          let o = {};
95          o.q = $('#MEM-q-'+i).val();
96          o.a = $('#MEM-a-'+i).val();
97          if(!o.q || o.q.length === 0 || !o.a || o.a.length === 0){
98              continue;
99          }
100         memQuestions.push(o);
101     }
```

Hier worden de vragen en antwoorden ophehaald gebruikmakende van Jquery. Het invoerveld van de vraag en het invoerveld van het antwoord worden geselecteerd en worden de waarde uitgelezen en in het object o gestopt. Daarna wordt er gekeken of de vraag en het antwoord gevonden zijn en of ze wel wat tekst in zich hebben. Als dat zo is wordt het object met de vraag en het antwoord in een array opgeslagen.

```
123     let dataObj = {
124         'mem': memQuestions,
125         'mc': mcQuestions,
126         'oq': oqQuestions,
127         'mcE': mcChecked,
128         'oqE': oqChecked,
129     }
130
131     let dataStr = btoa(JSON.stringify(dataObj));
132     copyStringToClipboard(dataStr);
133     alert('Import data copied to clipboard');
134 }
```

Vervolgens wordt deze array samen met de arrays met vraag en antwoorden van de andere minigames in een ander overkoepelend object gestopt. Dit object wordt dan een Json string van gemaakt en met Base64 versleuteld.




Als laatst wordt deze datastring automatisch gekopieerd naar het clipboard waarna de gebruiker deze in de game kan plakken en importen.

## Het spel

In het spel is het mogelijk de vragen uit de editor te importen waarna het spel start en de vragen ingeladen worden in het memory spel en de verschillende minigames. Vervolgens krijg je 30 seconden de tijd om memory te spelen waarna de game switcht naar een random minigame (tenzij er maar 1 minigame in de editor aan staat). Nu heb je 20 seconden de tijd om de minigame te spelen waarbij aan het eind van de 20 seconden er 1 seconde bij de 30 seconden van de memory tijd bij geteld word voor elk goed beantwoorde vraag in de minigame.

### Importen

Wanneer de gebruiker klaar is met vragen aanpassen in de editor kan de gebruiker de vragen en antwoorden exporteren en importeren in het spel.

We nemen als voorbeeld het importeren van de memory vragen en antwoorden. (de code is terug te vinden in js/game.js en js/memory.js)

```
71  function onGameImport(){
72      let str = $('#import-textarea').val();
73      let data = JSON.parse(atob(str));
74
75      memoryGame = new Memory(this, data['mem']);
76
77      if(data['mcE']){
78          games.push(new MultipleChoice(data['mc']));
79      }
80      if(data['oqE']){
81          games.push(new OpenQuestions(data['oq']));
82      }
83
84      startGame();
85  }
```

Wanneer er op de import knop gedrukt is wordt de tekst in het invoerveld eerst met Base64 ontsleuteld en vervolgens van Json string naar object omgezet. Dan wordt er een nieuwe instantie gemaakt van het memory spel en de array van vragen en antwoorden wordt uit het dataobject gehaald en meegegeven.

```
2       constructor(game, data) {
3           this.game = game;
4           this.questions = [];
5           for(let i = 0; i < data.length; i++){
6               this.questions.push({id: i, question: data[i].q, answer: data[i].a});
7           }
8           this.guessCount = 0;
9       }
```

De vraag/antwoord array wordt vervolgens uitgelezen en op een formaat opgeslagen in de Memory klasse waarnaar later gerefereerd kan worden bij het aanmaken van de memory kaartjes.