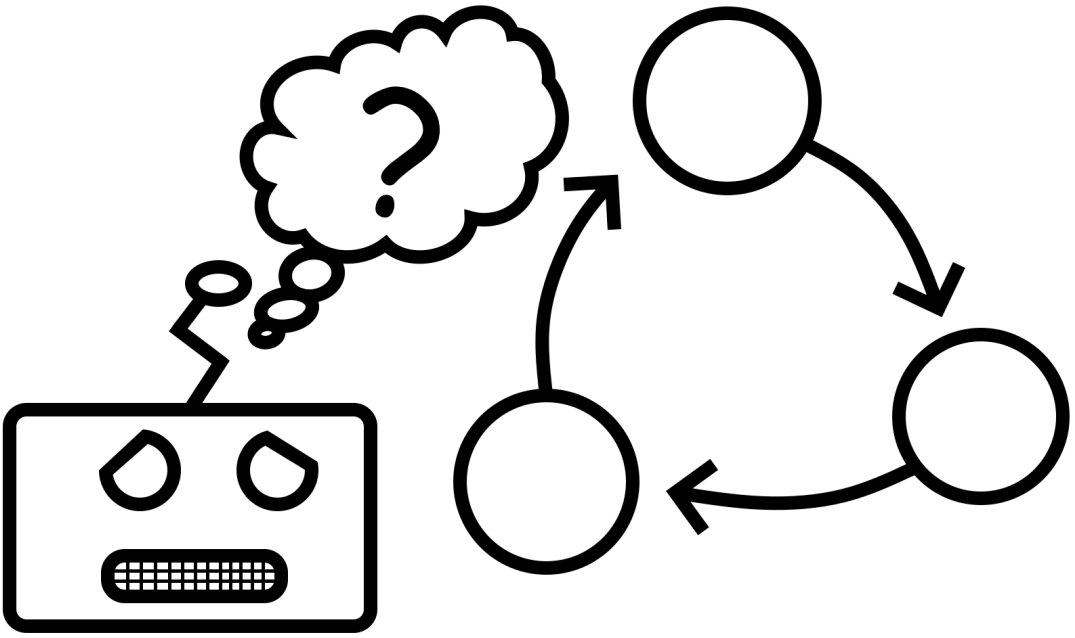
# Model-free Prediction & Control

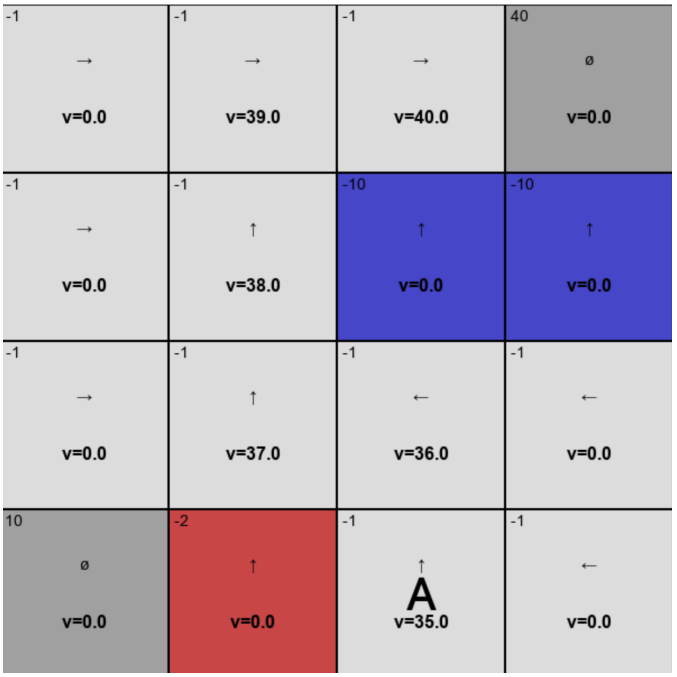
Programmeeropdracht



John Williams - 1790472

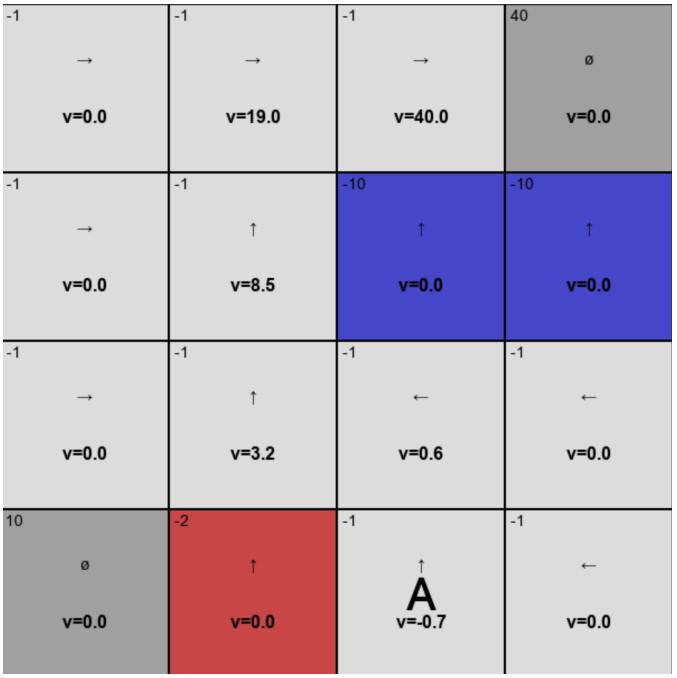
# A. Temporal Difference Learning

I. *Implementeer temporal difference learning. Voer de evaluatie uit op de optimale policy π∗ met γ = 1 en γ = 0.5. Visualiseer de uitkomsten en verklaar het resultaat.*




Hier eerst de uitvoering van de evaluatie met γ = 1. Via de pijlen is de optimale policy π* te zien. Wat opvalt is dat alleen de states direct in het pad van de optimale policy een veranderde waarde hebben (tegenover de initialisatiewaarde 0). Deze waarden verschillen ieder met 1 vergeleken met de volgende/vorige state die door de policy bezocht wordt. De state naast de terminal rechtsboven verwacht de volle reward van de terminal state, terwijl iedere state daarvoor steeds 1 minder in hun reward verwachten, omdat de states zelf een reward van -1 hebben.

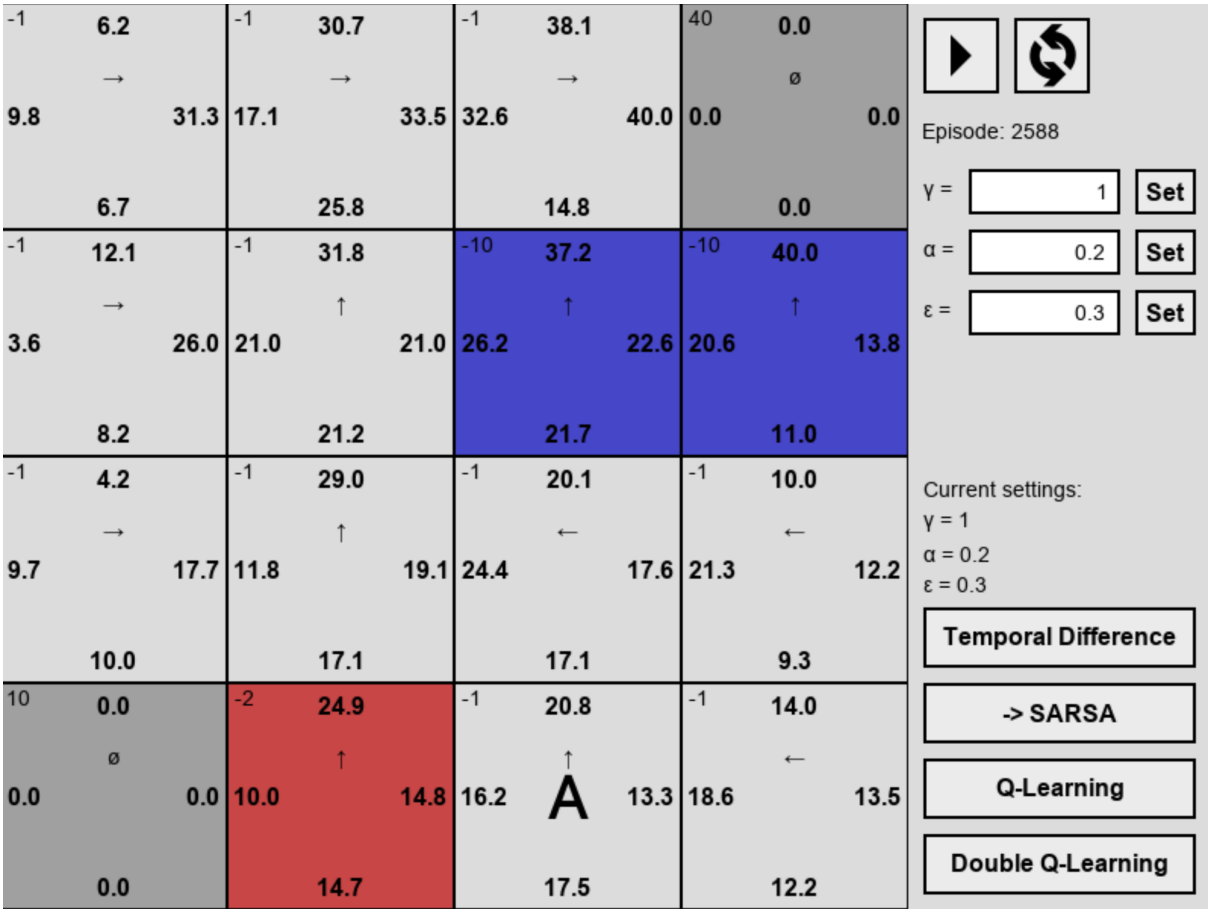
Met een discount van 1 zijn alle toekomstige stappen even belangrijk, dus vandaar dat bij de startpositie al een hoge reward verwacht wordt omdat het een paar stappen van de rerward van 40 af is.



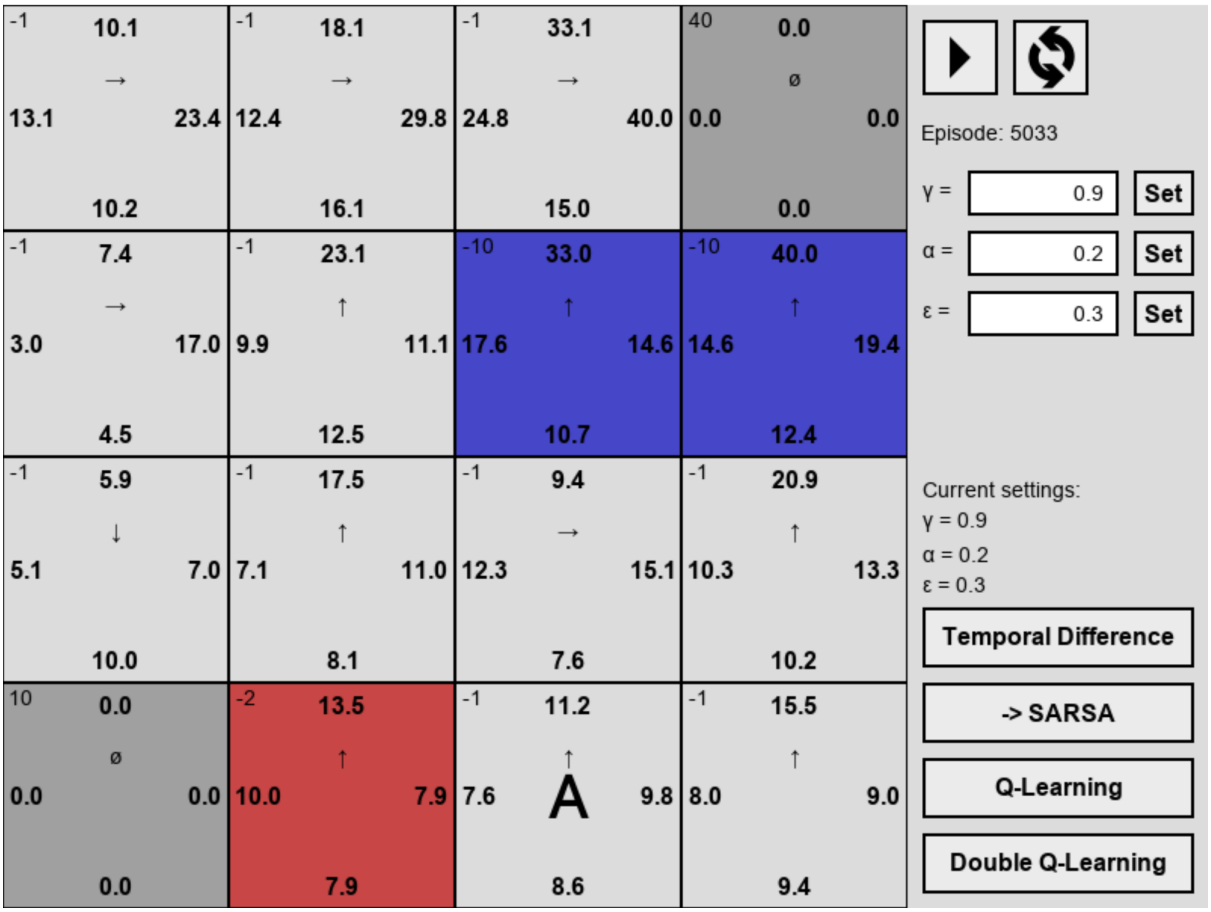

Wanneer een discount γ = 0.5 wordt gebruikt is het veel duidelijker dat in de toekomst kijken niet veel waarde heeft. De aantallen waarmee de verwachte reward daalt is significant, tot het punt dat de verwachte reward van de startpositie slechter is dan hoe het geïnitialiseerd was.

# B. SARSA

I. *Implementeer SARSA (on-policy TD control). Voer control uit met γ = 1 en γ = 0.9, visualiseer de uitkomsten en verklaar het resultaat.*



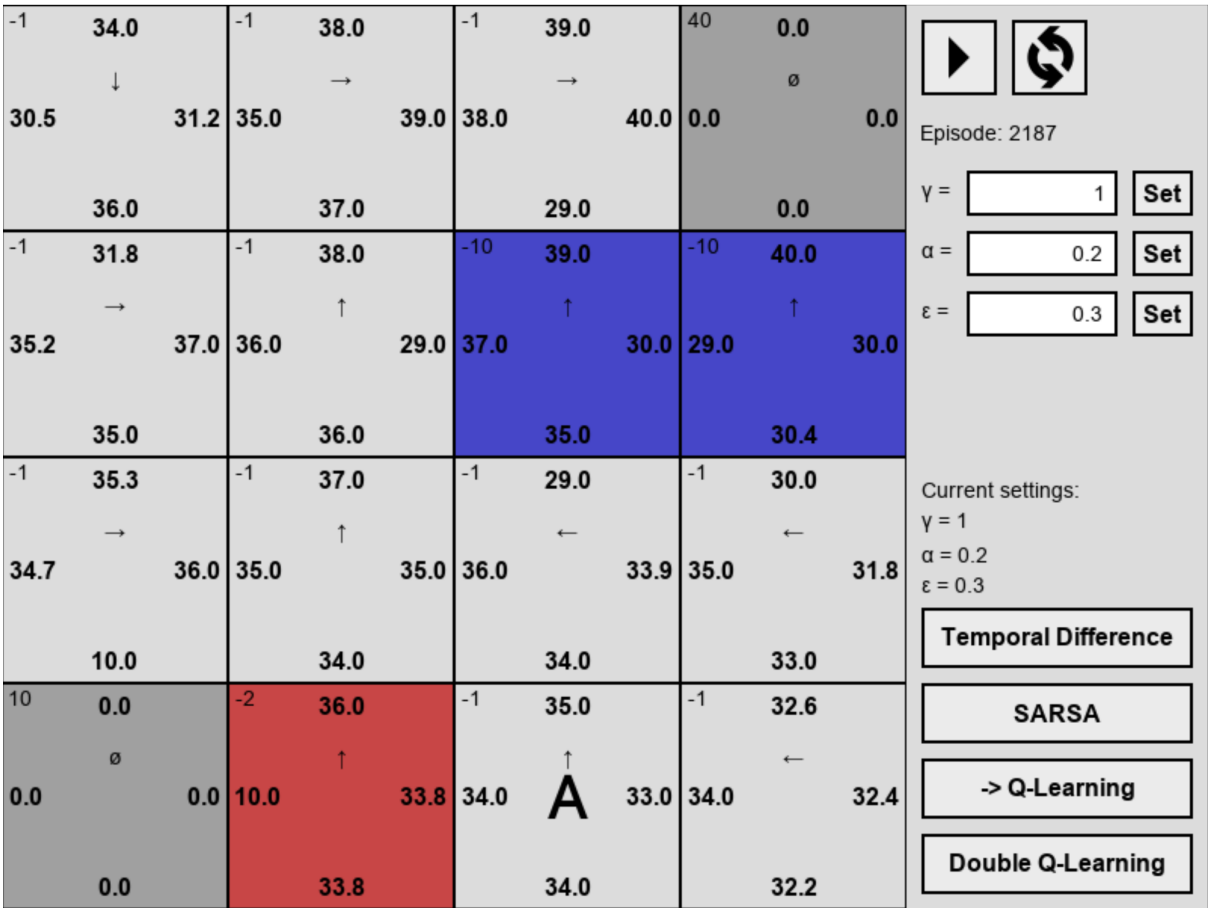

Hier, waarbij een discount 1 gebruikt is, is te zien dat de policy zich tot de optimale policy vormt. Er is gekozen voor een learning rate van 0.2, om beter de optimale waarden te kunnen benaderen. Er is ook gekozen voor een exploration rate van 0.3. Wat opvalt is dat de agent aan het begin heel graag via de rode state naar de terminal state linksonder gaat. Dit is namelijk een pad die heel makkelijk te bereiken is vanaf het beginpunt. Echter na vele episodes wordt de agent toch consistent in het bereiken van de terminal state rechtsboven.



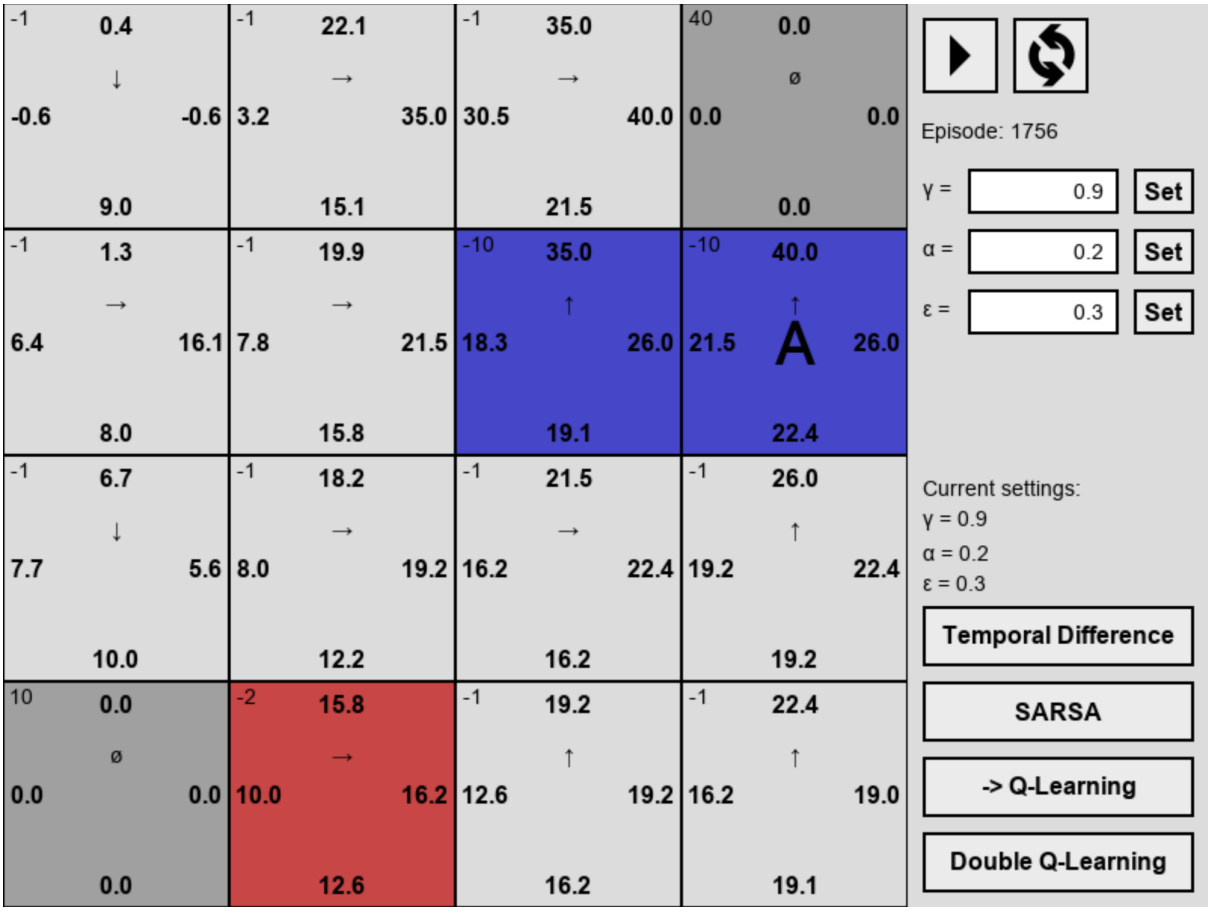

Hier wordt een discount gebruikt van 0.9, en na dubbel zo veel episodes als de situatie hierboven is het nog steeds niet consistent in het volgen van de optimal policy. De agent heeft de neiging om toch door de blauwe states te gaan, ten kostte van een hogere reward.

# C. Q-Learning

I. *Implementeer Q-learning (off-policy TD control). Voer control uit met γ = 1 en γ = 0.9, visualiseer de uitkomsten en verklaar het resultaat.*



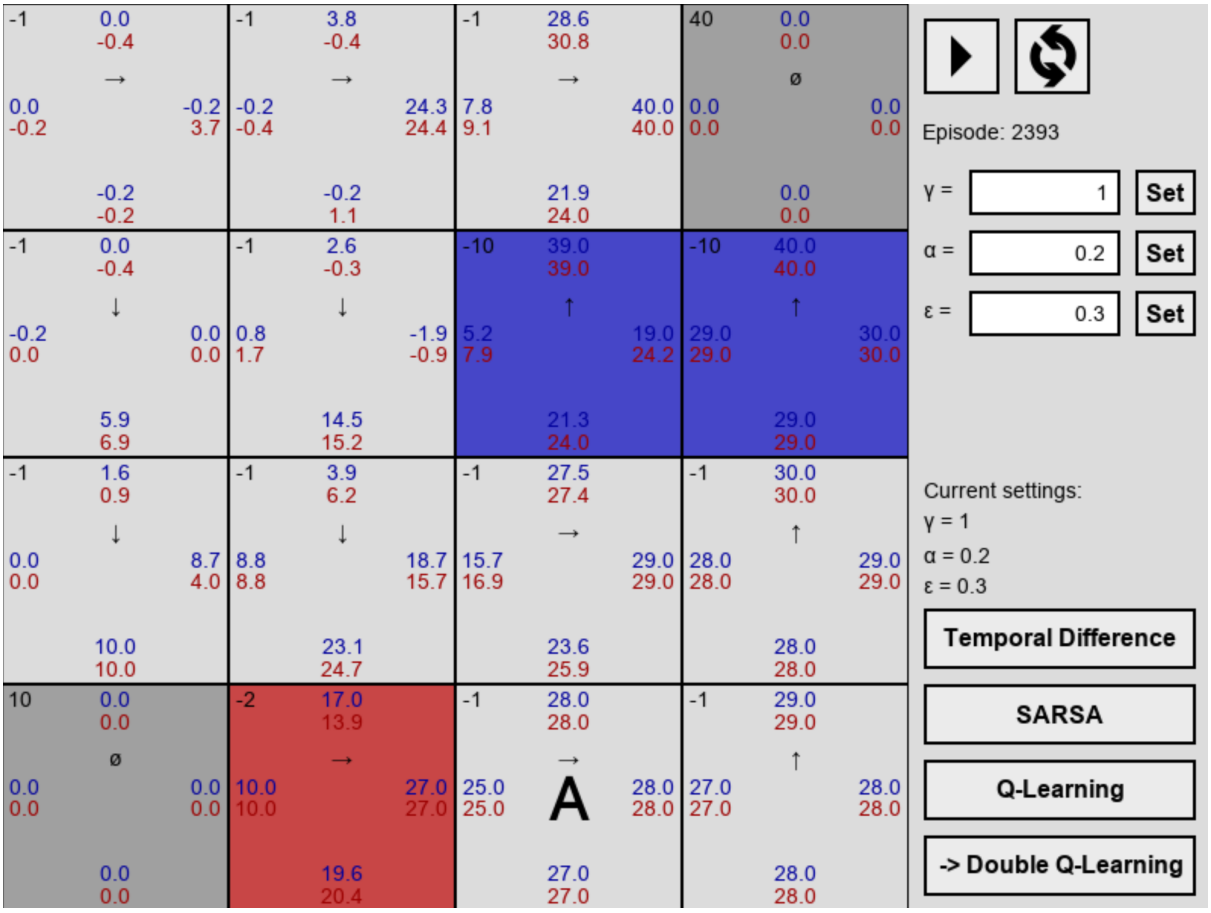

Hier is Q-Learning uitgevoerd met een discount van 1, en wat meteen opvalt is dat de values van de acties die de optimal policy geconvergeerd zijn tot die van de optimal policy zelf. Dit verschilt met SARSA omdat SARSA een actie kiest van de volgende state op een ε-greedy manier, terwijl Q-Learning altijd de beste actie van de volgende state kiest. Dit betekent dat wanneer het de optimale actie kiest, dat die waarde geüpdate wordt met behulp van de optimale actie in de volgende state. SARSA update deze waarde met mogelijk de optimale, maar ook mogelijk een suboptimale. Hierdoor heeft SARSA meer moeite met convergeren dan Q-Learning.



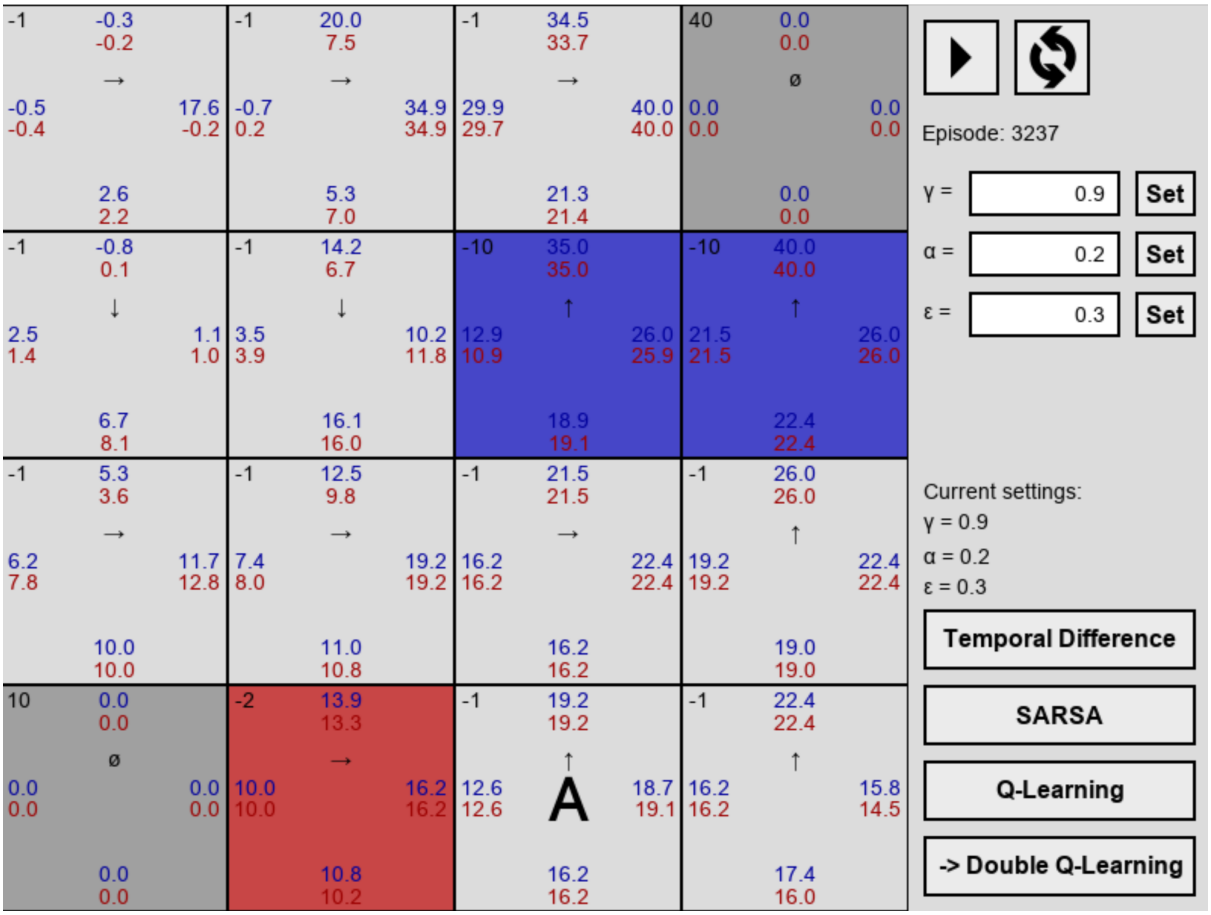

Hier, met een discount van 0.9, is de policy geconvergeerd tot een andere dan de optimal, namelijk eentje die de voorkeur heeft om via de blauwe states te gaan. De waardes veranderen ook niet meer, dus dit is met een discounted vooruitzicht het beste wat je gaat krijgen.

# E. Double Q-Learning

I. *Implementeer Double Q-learning. Voer control uit met γ = 1 en γ = 0.9, visualiseer de uitkomsten en verklaar het resultaat. Vergelijk de resultaten tenslotte met de resultaten van Q-learning.*




Hier, met een discount van 1, wordt Double Q-Learning uitgevoerd. Wat meteen opvalt vergeleken met de reguliere Q-Learning met een discount van 1, is dat Double Q-Learning niet heeft geconvergeerd tot de optimale policy. Hoe dit gebeurt is kan ik mijzelf momenteel niet verklaren.




Hier is Double Q-Learning uitgevoerd met een discount van 0.9, en net als bij een discount van 0.9 is de policy niet geconvergeerd tot de optimale policy. Deze lijkt echter wel op zijn gelijke bij de reguliere Q-Learning.