Model.Main

Main

+database: snakeDAO

+showMainMenu()
+setGameConfig()
+setSimulationConfig()
+showDiagrams()
+quit()
+startGame(String headColor, String bodyColor, String appleColor, int speed, int boardSize)
+startSimulation(String headColor, String bodyColor, String appleColor, int boardSize)

GraphWindow

+database: SnakeDAO
+graphBuilder: GraphBuilder

+showGraphWindow()
+showEpisodeSnakePoints()
+showEpisodeReward()
+showEpisodeTime()
+showEpisodeSteps()
+quit()

GraphBuilder

+buildGraph(String yAxisLabel, String title, String legend, list: int xValues, list: int yValues)

DAO

+saveEpisode()
+getEpisodes(): list<int>
+getEpisodeSnakePoints(): list<int>
+getEpisodeReward(): list<int>
+getEpisodeTime(): list<int>
+getEpisodeSteps(): list<int>
+databaseConnection()
+saveColor()
+saveHighscore()
+getHighscore()

SnakeDAO

+saveEpisode()
+getEpisodes(): list<int>
+getEpisodeSnakePoints(): list<int>
+getEpisodeReward(): list<int>
+getEpisodeTime(): list<int>
+getEpisodeSteps(): list<int>
+databaseConnection()
+saveColor()
+saveHighscore()
+getHighscore()

Game

+score: int
+highScore: int
+board: Board
+database: SnakeDAO

+showGame()
+pause()
+resume()
+quit()
+addPoints()
+reset()

Simulation

+board: Board
+agent: Agent

+showSimulation()
+deepQLearning()
+pauseAll()
+resumeAll()
+saveMemory()

Board

+boardSize: int
+snake: Snake
+apple: Apple
+boardList: list<char>
+game: Game

+addApple()
+snakeCollision()
+tic()
+isGameOver(): bool
+removeApple()

Agent

+bufferSize: int
+batchSize: int
+gamma: float
+tau: float
+learningRate: float
+updateNetwork: int
+stateSize: int
+actionSize: int
+seed: int
+qnetworkLocal: QNetwork
+qnetworkTarget: QNetwork
+agentMemory: ReplayBuffer

+step(list: char state, int action, float reward, list: char nextState, bool done)
+act(list: char state, float epsilon)
+learn(tuple experiences, float gamma)
+targetQNetUpdate(QNetwork localModel, QNetwork targetModel, int tau)

QNetwork

+forward(list: char state)

Snake

+headColor: Color
+bodyColor: Color
+body: Body
+currentDirection: String
+snakeController: SnakeController
+speed: int

+tic()
+moveUp()
+moveDown()
+moveLeft()
+moveRight()
+grow()

Apple

+appleColor: Color
+point: Index

ReplayBuffer

+addExperience(list: char state, int action, float reward, list: char nextState, bool done)
+sampleExperience()

Color

+darkGreen
+lightGreen
+red
+yellow
+orange
+purple
+pink
+blue
+darkBlue
+white

Index

+x: int
+y: int

SnakeController

+snake: Snake

+checkPlayerInput()
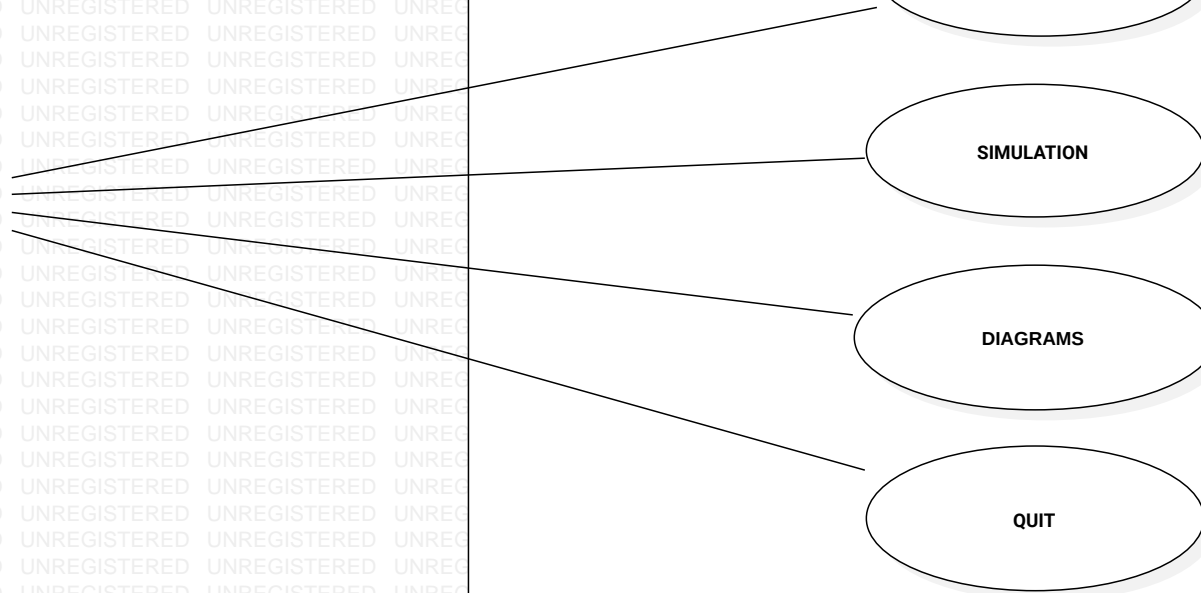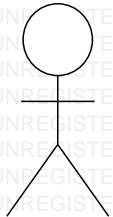+changeSnakeDirection(String currentDirection)

Body

+append()
+rotate()

S N A.I. K E

PLAY

SIMULATION

DIAGRAMS

QUIT

User

CONFIGURATION JEU

Head

Set colors

Body

Apple

Set snake speed

Set board size

User

START GAME

CANCEL

Model Cas Usage JEU

JEU
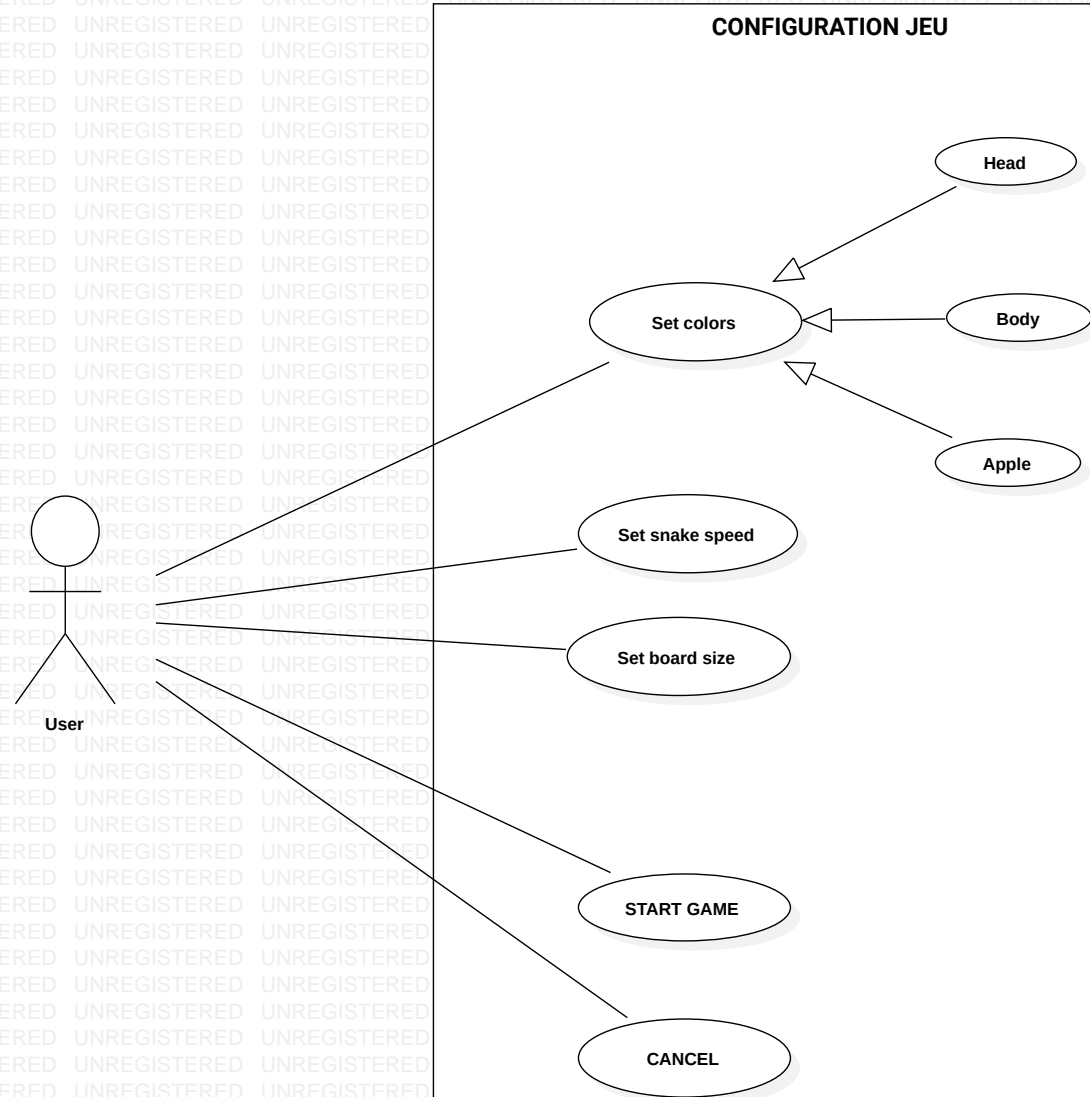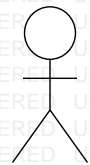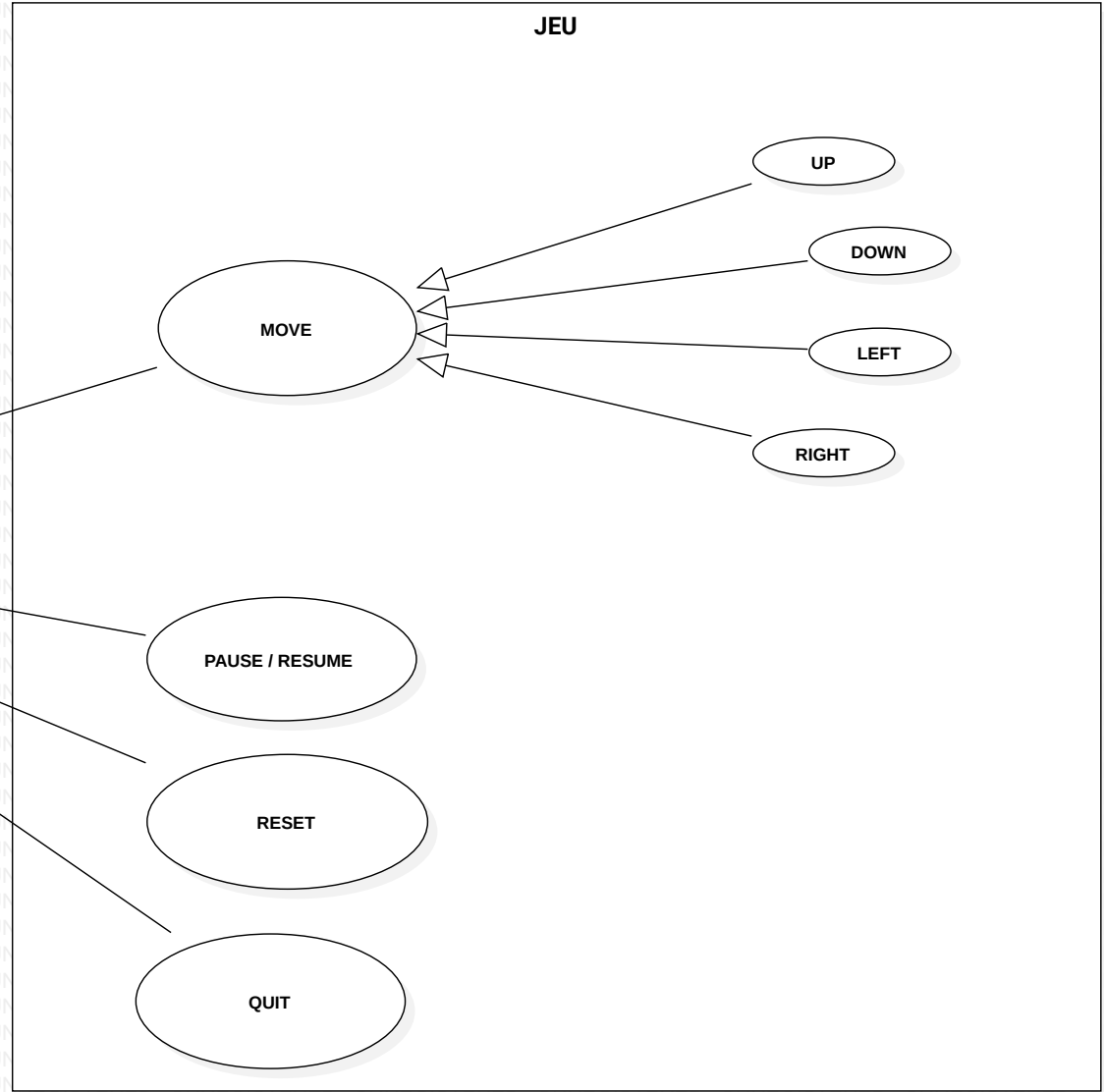
CONFIGURATION SIMULATION
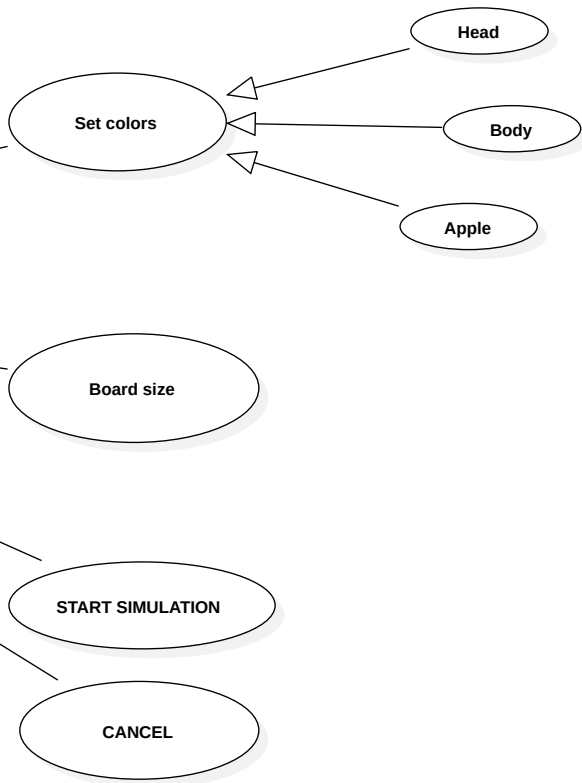
Head

Set colors

Body

Apple

User

Board size

START SIMULATION

CANCEL
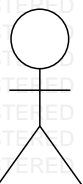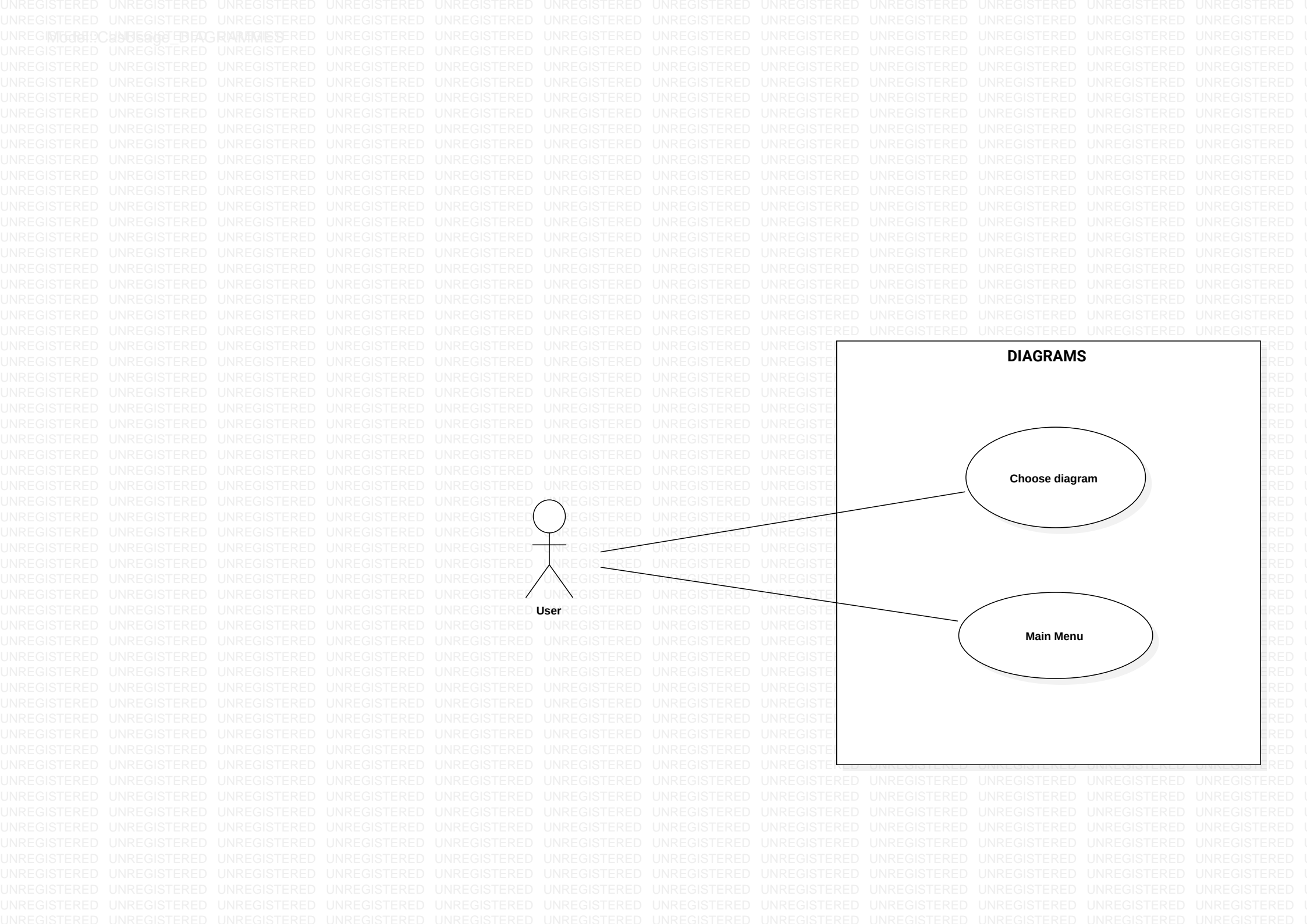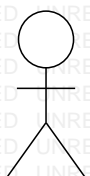
DIAGRAMS

Choose diagram

Main Menu

User

MODEL DB Tables DB
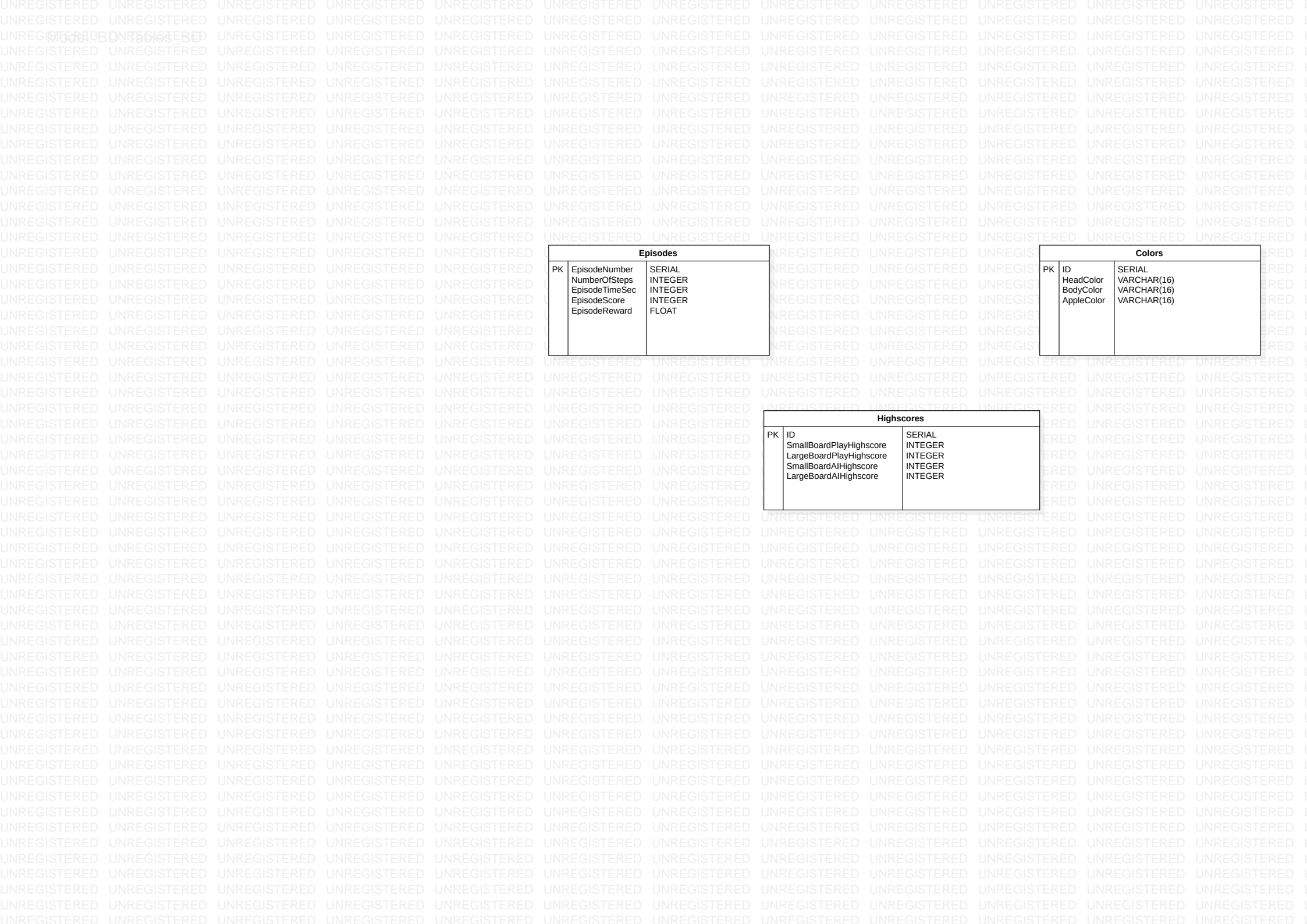
Episodes

PK	EpisodeNumber	SERIAL
	NumberOfSteps	INTEGER
	EpisodeTimeSec	INTEGER
	EpisodeScore	INTEGER
	EpisodeReward	FLOAT

Colors

PK	ID	SERIAL
	HeadColor	VARCHAR(16)
	BodyColor	VARCHAR(16)
	AppleColor	VARCHAR(16)

Highscores

PK	ID	SERIAL
	SmallBoardPlayHighscore	INTEGER
	LargeBoardPlayHighscore	INTEGER
	SmallBoardAIHighscore	INTEGER
	LargeBoardAIHighscore	INTEGER

sd Visualisation_Graphes

GraphWindow	SnakeDAO	Database	GraphBuilder

1 : showDiagrams()

2 : Choix du graphique

3 : getInfo()

4 :

5 :

6 : showGraph()

7 : buildGraph()

Modèle Dynamique : Sequence_Jeu-Jeu

sd Jeu

Participants: **Board**, **Snake**, **Apple**, **Game**, **SnakeDAO**

1 : tic()
2 : tic()
3 :
4 : tic()
5 :
6 : snakeCollision

alt AppleCollision
[if collision with apple]
7 : Grow
8 : grow()
9 : addApple()

alt GameOver
[if collision with wall or tail]
10 : Game Over

alt Highscore
[if score > highscore]
11 : Save highscore
12 : saveHighscore()
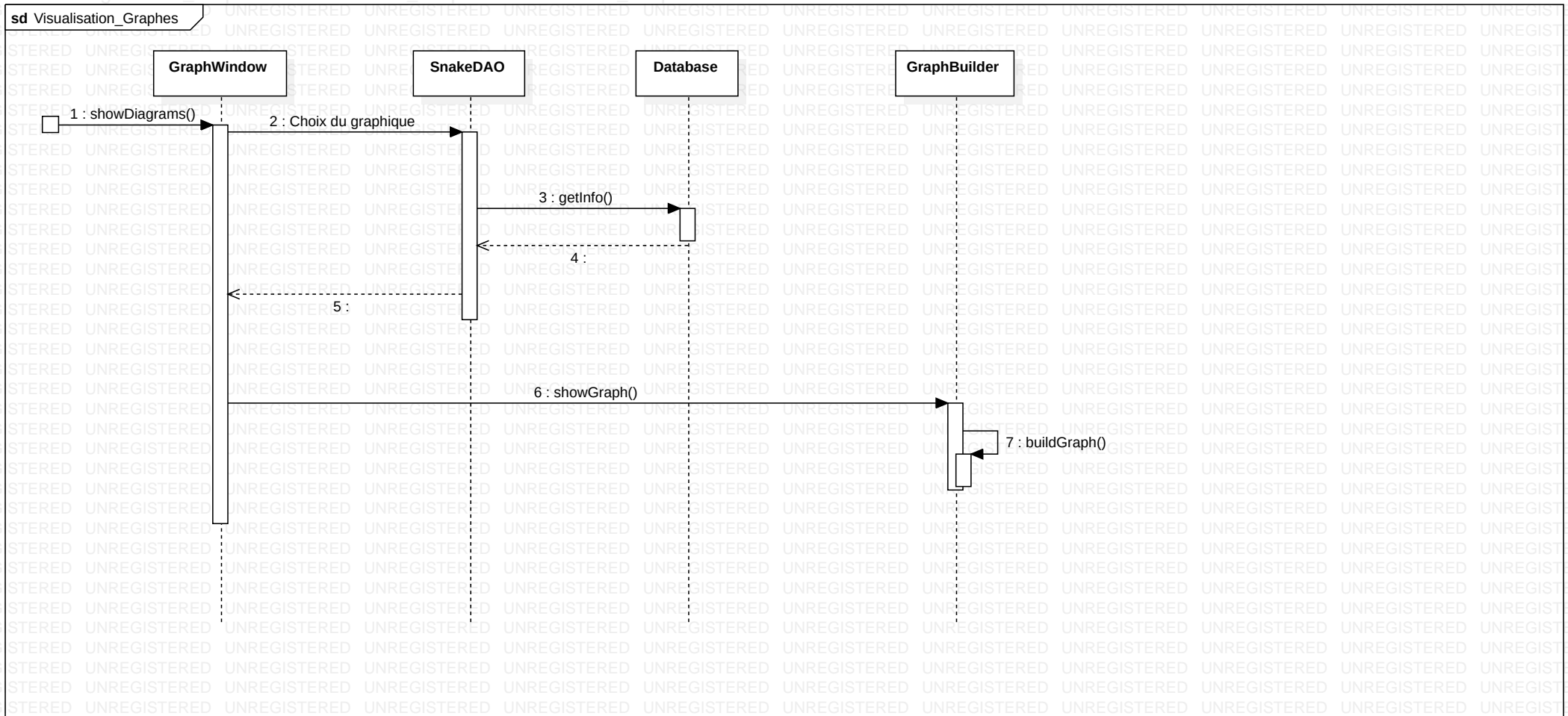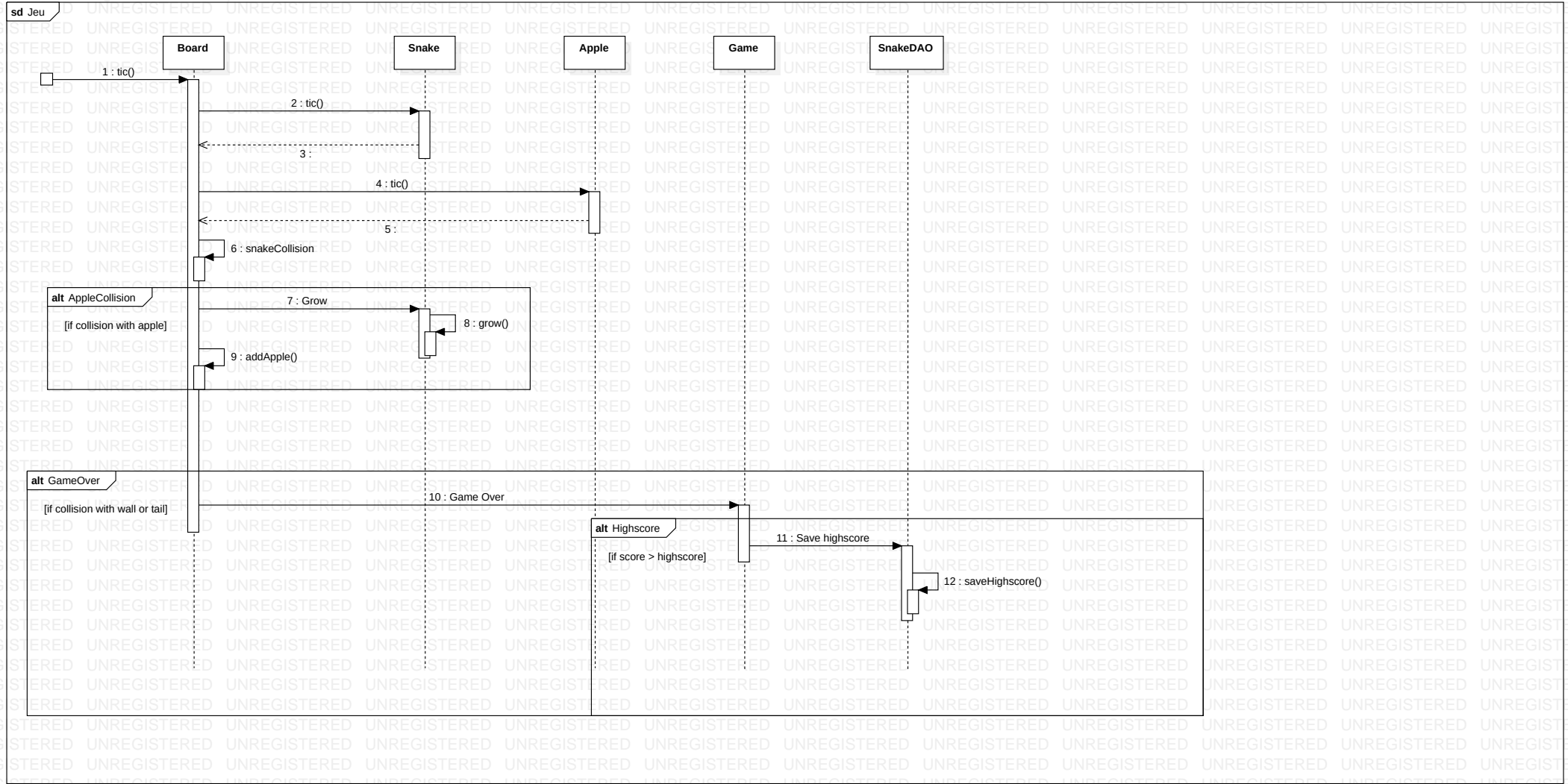
sd Simulation

Simulation

Agent

1 : tic()

2 : Action to take

3 : act()

4 :

5 : Make a step

6 : step()

alt Save memory

7 : saveMemory()

[if episode number == 100]