Rapport ADA

— EPITA 2021 —

Florent Carrez florent.carrez@epita.fr
Alice Goudout alice.goudout@epita.fr
Raphaël Treglia raphael.treglia@epita.fr

Table des matières

Chapitre 1

Introduction

1.1 Contexte

Ce projet vise à implémenter un jeu de type Snake en langage ADA. Il est réalisé en natif faute d'avoir pu se procurer le matériel adéquate en ces temps de conditions sanitaires strictes.

Nous sommes donc contraints à programmer en ADA sur un ordinateur et nous avons donc seulement un écran pour interagir avec l'utilisateur.

Étant donné le fait que le projet allait être simplifié dans son architecture globale, nous avons du nous concentrer sur le fait d'essayer de proposer un snake un peu différent avec des fonctionnalités supplémentaires dont on parlera dans ce rapport.

1.2 Lancer le projet

Pour exécuter et tester le jeu, il n'y a pas beaucoup de pré-requis. Il suffit d'avoir une version récente de la suite GNAT permettant de compiler et d'exécuter du code ADA.

Pour commencer, on peut trouver le code dans un répertoire github publique à l'adresse suivante : https://github.com/DarkMiMolle/AdaSnake

On peut y retrouver une documentation sous format Markdown comme demandé, ainsi que la liste des features et des exemples de codes.

Pour compiler, il suffira d'aller dans le répertoire principal et de lancer la commande :

Build with `gprbuild`:

```
gprbuild -d -PSnake/snake.gpr Snake/src/main.adb
```

Run:

```
./Snake/obj/main
```

Chapitre 2

DO178C

2.1 Processus de développement

2.1.1 Exigences de haut niveau (HLR)

- **REQ.1 Gagner**
  Lorsque le joueur remplit tout le terrain, le jeu s'arrête et lui annonce qu'il a gagné en affichant son score.
- **REQ.2 Perdre**
  Lorsque le joueur perd, le jeu s'arrête en lui annonçant qu'il a perdu en affichant son score.
- **REQ.3 Affichage du terrain**
  Les limites du terrain à ne pas dépasser par le joueur s'affichent correctement.
- **REQ.4 Apparition des jetons**
  L'apparition des jetons a toujours lieu dans un endroit accessible par le joueur.
- **REQ.5 Position du joueur**
  Le joueur est toujours à l'intérieur du terrain tracé à l'écran.
- **REQ.6 Consommer les jetons**
  Lorsque la tête du personnage joueur passe sur un jeton, celui-ci disparaît.
- **REQ.7 Augmentation du score**
  Lorsque la tête du personnage joueur passe sur un jeton, son score augmente.
- **REQ.8 Augmentation de la taille**
  Lorsque la tête du personnage joueur passe sur un jeton, un élément est rajouté au corps du joueur.
- **REQ.9 Touches**
  Lorsque le joueur appuie sur une touche directionnelle, il se déplace dans la direction indiquée.
- **REQ.10 Positionnement du corps**
  Le positionnement du corps du joueur est correct selon les règles du Snake.
- **REQ.11 Zoom**
  L'option de zoom fonctionne correctement.

- **REQ.12 Couleurs**
  La couleur sélectionnée s'applique au niveau.

2.1.2 Architecture globale et logicielle

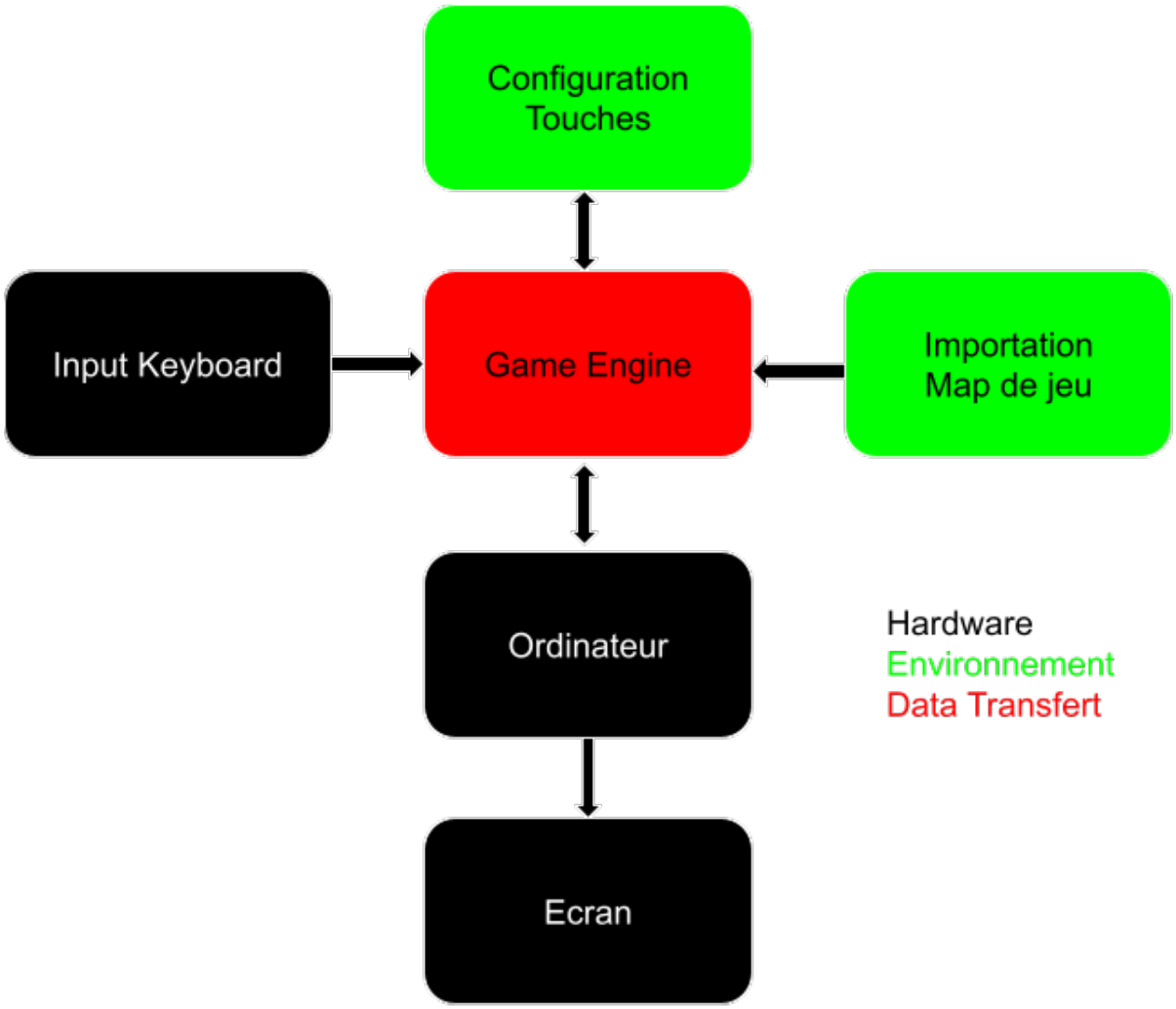

FIGURE 2.1 – Architecture globale

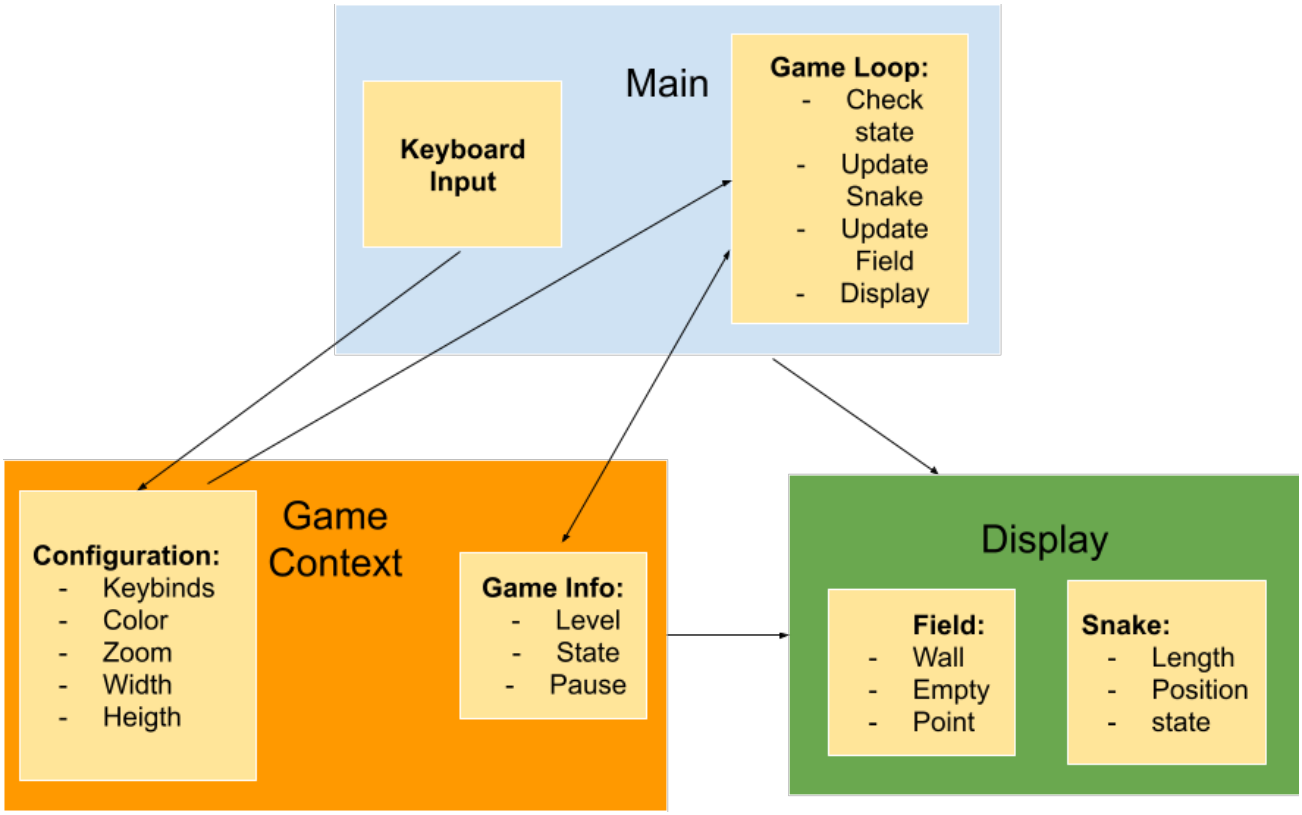

FIGURE 2.2 – Architecture logicielle

2.1.3 Exigences de bas niveau (LLR)

- **REQ.1.1 Taille max**
  La taille maximum du vecteur du serpent est atteinte.
- **REQ.1.2 Partie terminée**
  Lorsque la partie est finie, le statut de la partie est "terminé".
- **REQ.2.1 Collision**
  Une collision arrête la partie.
- **REQ.3.1 Affichage terrain**
  Lors de l'affichage, les bords du terrain sont symbolisés par des '#'.
- **REQ.4.1 Affichage jeton**
  Les jetons s'affichent à leur position.
- **REQ.4.2 Position jeton**
  La position des jetons est générée aléatoirement.
- **REQ.5.1 Affichage joueur**
  Le joueur s'affiche correctement.
- **REQ.7.1 Gagner des points**
  Le score est mis à jour en même temps que l'incrémentation du nombre d'éléments du corps du joueur.

- **REQ.8.1Grandir**
  Le vecteur de positions des éléments du corps est incrémenté de un lorsque le joueur passe sur un jeton.
- **REQ.9.1 Possibilité déplacement**
  Un case réparti les informations reçues du clavier.
- **REQ.11.1 Zoom +**
  Vérifie si zoomer est possible.
- **REQ.11.2 Zoom -**
  Vérifie si dézoomer est possible.
- **REQ.12.1 Disponibilité couleur** Une couleur doit être sélectionnée pour s'appliquer.

2.1.4 Méthodes de traçabilité

La traçabilité sera assurée par des tableaux récapitulatifs qui feront le lien entre HLR, LLR et tests.

HLR	LLR
REQ.1 Gagner	REQ.1.1 Taille max REQ.1.2 Partie terminée
REQ.2 Perdre	REQ.2.1 Collision
REQ.3 Affichage du terrain	REQ.3.1 Affichage terrain
REQ.4 Apparition des jetons	REQ.4.1 Affichage jeton REQ.4.2 Position jeton
REQ.5 Position du joueur	REQ.5.1 Affichage joueur
REQ.7 Augmentation du score	REQ.7.1 Gagner des points
REQ.8 Augmentation de la taille	REQ.8.1 Grandir
REQ.9 Déplacement	REQ.9.1 Possibilité déplacement
REQ.11 Zoom	REQ.11.1 Zoom + REQ.11.2 Zoom -
REQ.12 Couleurs	REQ.12.1 Disponibilité couleur

FIGURE 2.3 – Traçabilité entre HLR et LLR

2.2 Processus de vérification

2.2.1 Cas de test de vérification logicielle et des procédures pour HLR et LLR

Cas de test HLR

Si un test n'est pas rempli, l'erreur correspondante est renvoyée.

- **TC.1.1 Remplissage**
  Le joueur peut remplir tout le terrain.
- **TC.1.2 Affichage**
  Le score s'affiche lorsque le joueur termine sa partie.
- **TC.2.1 Toucher un mur**
  Toucher un mur met fin à la partie.
- **TC.2.2 Toucher son corps**
  Toucher son corps met fin à la partie.
- **TC.3.1 Terrain**
  Le terrain reste correctement affiché tout au long de la partie.
- **TC.4.1 Position jeton**
  La position du jeton est inférieure aux maximums de la hauteur et de la largeur du terrain.
- **TC.5.1 Position joueur**
  Toutes les positions par lesquelles passe la tête du joueur sont inférieures aux maximums de la hauteur et de la largeur du terrain.
- **TC.6.1 Disparition visible**
  Le points disparaît quand la tête du serpent passe dessus
- **TC.6.2 Repasser**
  Si la tête repasse à la même position après la disparition du point, il ne se passe rien
- **TC.7.1 Augmentation du score**
  Lorsque la tête passe sur un jeton, le score du joueur est augmenté de 1.
- **TC.8.1 Rajout**
  N+1 éléments du corps apparaissent à l'écran.
- **TC.9.1 Haut**
  Appuyer sur 'z' fait avancer d'une unité vers le haut.
- **TC.9.2 Bas**
  Appuyer sur 's' fait avancer d'une unité vers le bas.
- **TC.9.3 Droite**
  Appuyer sur 'd' fait avancer d'une unité vers la droite.
- **TC.9.4 Gauche**
  Appuyer sur 'q' fait avancer d'une unité vers la gauche.
- **TC.9.5 Pause**
  Appuyer sur 'p' met le jeu en pause.

- **TC.9.6 Exit**
  Appuyer sur 'e' fait sortir du jeu.
- **TC.10.1 Positionnement**
  Les N éléments du corps du joueur occupent les N dernières positions de la tête du joueur.
- **TC.11.1 Zoom +**
  Lorsque le + du zoom est sélectionné, tous les éléments s'agrandissent.
- **TC.11.2 Zoom –**
  Lorsque le - du zoom est sélectionné, tous les éléments rétrécissent.
- **TC.12.1 Blue**
  Les éléments deviennent bleus lorsque Blue est sélectionné dans les couleurs.
- **TC.12.2 Red**
  Les éléments deviennent rouges lorsque Red est sélectionné dans les couleurs.
- **TC.12.3 Green**
  Les éléments deviennent verts lorsque Green est sélectionné dans les couleurs.
- **TC.12.4 Brown**
  Les éléments deviennent marrons lorsque Brown est sélectionné dans les couleurs.
- **TC.12.5 Black**
  Les éléments deviennent noirs lorsque Black est sélectionné dans les couleurs.
- **TC.12.6 None**
  Les éléments restent blancs lorsqu'aucune couleur n'est sélectionnée.

Cas de test LLR

- **TC.1.1.1 Taille max**
  Vérifie que la taille du vecteur fait maxHeight * maxWidth.
- **TC.1.2.1 Stop**
  Vérifie que GameStopedInfo est en "Stoped".
- **TC.2.1.1 Se rentrer dedans**
  Vérifie que GameStopedInfo a bien la valeur LostSnakeEatItself lorsque le joueur touche son corps.
- **TC.2.1.2 Rentrer dans un mur**
  Vérifie que GameStopedInfo a bien la valeur LostSnakeOnWall lorsque le joueur touche un mur.
- **TC.3.1.1 Terrain**
  Vérifie que le terrain affiche un rectangle de hauteur maxHeight et de largeur maxWidth.
- **TC.4.1.1 Coordonnées égales**
  Vérifie que l'affichage correspond aux coordonnées du jeton.
- **TC.4.2.1 Bonnes coordonnées**
  Vérifie que les coordonnées du nouveau jeton sont bien inférieures à la hauteur et la largeur du terrain.

- **TC.5.1.1 Eléments**
  Vérifie que le joueur est composé du bon nombre d'éléments, donc de sa taille initiale plus les jetons ramassés.
- **TC.5.1.2 Position**
  Vérifie la bonne position de tous les éléments du joueur.
- **TC.7.1.1 Score**
  Vérifie que le score est égal au nombre d'élements du corps du joueur.
- **TC.8.1.1 Grandir**
  Vérifie que Snake.elems.Length'Old + 1 = Snake.elems.Length lorsqu'un jeton est consommé.
- **TC.9.1.1 Perdu ?**
  Vérifie si le déplacement est possible dans la position demandée avant de l'effectuer.
- **TC.11.1.1 Zoomer**
  ZoomIndice est inférieur à 3.
- **TC.11.2.1 Dézoomer**
  ZoomIndice est supérieur à 1.
- **TC.12.1.1 Vérifier couleur**
  Vérifie que ctxt.conf.color est vrai.

2.2.2 Traces

HLR	Test Cases
REQ.1 Gagner	TC.1.1 Remplissage TC.1.2 Affichage
REQ.2 Perdre	TC.2.1 Toucher un mur TC.2.2 Toucher son corps
REQ.3 Affichage du terrain	TC.3.1 Terrain
REQ.4 Apparition des jetons	TC.4.1 Position jeton
REQ.5 Position du joueur	TC.5.1 Position joueur
REQ.6 Consommer les jetons	TC.6.1 Disparition visible TC.6.2 Repasser
REQ.7 Augmentation du score	TC.7.1 Augmentation du score
REQ.8 Augmentation de la taille	TC.8.1 Rajout
REQ.9 Déplacement	TC.9.1 Haut TC.9.2 Bas TC.9.3 Droite TC.9.1 Gauche TC.9.5 Pause TC.9.6 Sortir
REQ.10 Positionnement du corps	TC.10.1 Positionnement
REQ.11 Zoom	TC.11.1 Zoom + TC.11.2 Zoom –
REQ.12 Couleurs	TC.12.1 Blue TC.12.2 Red TC.12.3 Green TC.12.4 Brown TC.12.5 Black TC.12.6 None

FIGURE 2.4 – Traçabilité entre les HLR et les tests

LLR	Test Cases
REQ.1.1 Taille max	TC.1.1.1 Taille max
REQ.1.2 Partie terminée	TC.1.2.1 Stop
REQ.2.1 Collision	TC.2.1.1 Se rentrer dedans TC.2.1.2 Rentrer dans un mur
REQ.3.1 Affichage terrain	TC.3.1.1 Terrain
REQ.4.1 Affichage jeton	TC.4.1.1 Coordonnées égales
REQ.4.2 Position jeton	TC.4.2.1 Bonnes coordonnées
REQ.5.1 Affichage joueur	TC.5.1.1 Eléments TC.5.1.2 Position
REQ.7.1 Gagner des points	TC.7.1.1 Score
REQ.8.1 Grandir	TC.8.1.1 Grandir
REQ.9.1 Possibilité déplacement	TC.9.1.1 Perdu ?
REQ.11.1 Zoom +	TC.11.1.1 Zoomer
REQ.11.2 Zoom -	TC.11.2.1 Dézoomer
REQ.12.1 Disponibilité couleur	TC.12.1.1 Vérifier couleur

FIGURE 2.5 – Traçabilité entre les LLR et les tests

Chapitre 3

Programmation par contrats

```
4  package Field is
5
6          type Field is tagged private;
7
8          function G_CheckRepresentation(f: Field) return Boolean with Ghost;
9
10         function CreatField(ctxt: in out GameContext.Context) return Field
11                 with    Pre => ctxt'Unchecked_Access /= null,
12                                 Post => G_CheckRepresentation(CreatField'Result);
13
14         type FieldElem is (Empty, Wall, Space);
15         function Char(elem: in FieldElem) return Character;
16
17         function G_Context(f: in Field) return access GameContext.Context with Ghost;
18         function G_GameRunning(f: in Field) return Boolean with Ghost,
19                 Pre => f.G_Context /= null;
20         function G_GamePausing(f: in Field) return Boolean with Ghost,
21                 Pre => f.G_Context /= null;
22         function G_SnakePos(f: in Field) return Position with Ghost;
23         function G_PtPos(f: in Field) return Position with Ghost;
24         function G_FieldElemAt(f: Field; pos: Position) return FieldElem with Ghost;
25
26         function Check(f: in out Field; s: in out Snake.Snake) return Boolean
27                 with    Pre => f.G_GameRunning,
28                                 Post => f.G_GameRunning = (f.G_FieldElemAt(s.Pos) = Space) -- False = False --> True
29                         and ((f.G_GameRunning and then (f.G_FieldElemAt(f.G_PtPos) = Space and f.G_PtPos /= s.Pos)) or else not f.G_GameRunning);
30                 -- the last line says: if the game is running the point has moved, else the game is not running anymore
31         procedure Paint(f: in Field);
32         procedure DisplayPt(f: in Field)
33                 with    Pre => f.G_GameRunning;
34         procedure NextPoint(f: in out Field)
35                 with    Pre => f.G_GameRunning and (f.G_SnakePos = f.G_PtPos or (f.G_Context /= null and then f.G_Context.Game.Pausing)),
36                                 Post => f.G_PtPos'Old /= f.G_PtPos and f.G_FieldElemAt(f.G_PtPos) = Space and f.G_PtPos /= f.G_SnakePos;
37         procedure HidePt(f: in Field)
38                 with    Pre => f.G_GamePausing;
39         -- pre: game pausing
```

FIGURE 3.1 – Fichier **field.ads**

```
function Running(g: in GameInfo) return Boolean;
function Pausing(g: in GameInfo) return Boolean
        with    Pre => g.Running;
procedure StopGame(g: in out GameInfo; reason: GameStopedInfo)
        with    Pre => g.Running,
                        Post => not g.Running;
procedure Pause(g: in out GameInfo)
        with    Pre => g.Running,
                        Post => g.Running and not g.Pausing;


-- Context
function CreatContext(width, height: SizeTerm) return Context;
-- Pre => abs(width - height) < 20
-- Post => return.MaxWidth == width, ..., return.Game.Running == true

function MaxWidth(ctxt: in Context) return SizeTerm;
function MaxHeight(ctxt: in Context) return SizeTerm;
function Config(ctxt: in Context) return Configuration'Class; -- to make the dispatching possible
function Game(ctxt: in out Context) return access GameInfo'Class
        with Post => ctxt.Game /= null; -- to make the dispatching possible
-- return *GameInfo, we want to be able to modify it
-- not: to return the access to the GameInfo, we must use: 'Access or 'Unchecked_Access
-- the in out allows us to return a non const access.

function G_Game(ctxt: Context) return GameInfo'Class with Ghost;
procedure EndGame(ctxt: in Context; score: Integer)
        with    Pre => not ctxt.G_Game.Running;
```

FIGURE 3.2 – Fichier **gamecontext.ads**

```
package Snake is

	type Snake is tagged private;

	package Direction is
		type Dir is new Integer range -2 .. 2 with Static_Predicate => Dir /= 0;
		Up: constant Dir := -1;
		Down: constant Dir := 1;
		Left: constant Dir := -2;
		Right: constant Dir := 2;
	end Direction;
	function NextPosFrom(dir: in Direction.Dir; pos: Position) return Position
		with	Post => pos /= NextPosFrom'Result;


	function G_GameRunning(s: in Snake) return Boolean with Ghost;
	function G_Dir(s: in Snake) return Direction.Dir with Ghost;
	function G_Score(s: Snake) return Integer with Ghost;


	function Creat(ctxt: in out GameContext.Context) return Snake
		with	Pre => ctxt.Game.Running and ctxt'Unchecked_Access /= null;

	procedure Display(s: in Snake)
		with	Pre => s.G_GameRunning;

	procedure Move(s: in out Snake)
		with	Pre => s.G_GameRunning,
				Post => s.Pos'Old /= s.Pos and s.Pos = NextPosFrom(s.G_Dir, s.Pos);
	procedure Pos(s: in out Snake; p: Position)
		with	Post => s.Pos = p;
	function Pos(s: in Snake) return Position;
	procedure ChangeDir(s: in out Snake; dir: in Direction.Dir)
		with	Post => Integer(s.G_Dir) = Integer(dir);

	procedure AddPoint(s: in out Snake)
		with	Post => s.G_GameRunning and s.Score = s'Old.G_Score + 1;
	function Score(s: in out Snake) return Integer;
```

FIGURE 3.3 – Fichier **snake.ads**

Chapitre 4

Conclusion

A ce jour, notre projet permet de jouer normalement au jeu Snake, même si des améliorations peuvent évidemment toujours être apportées. Nous n'avons pas pu implémenter toutes les features que nous voulions mais nous avons essayé de faire de notre mieux compte tenu de cette période de transition en stage.

Notre seul regret est de ne pas avoir eu accès à plus de matériel et de ressources en général, car cela aurait été plus intéressant. Nous comprenons les restrictions liées à ces temps de pandémie et avons fait du mieux possible pour s'adapter et pouvoir travailler correctement en groupe.

Nous avons quand même pu apprendre beaucoup de choses concernant le langage ADA, ainsi que la programmation par contrat, car c'était la première fois que nous codions plus que quelques lignes. La certification DO178C nous a aussi fait une bonne introduction en la matière malgré le fait que nous ayons fait seulement un jeu Snake qui ne correspond pas vraiment à de l'aviation civile.

Table des figures