Sciper	Archi (2pts)	architecture violation comment	Class etc... (3pts)	class violation comment
329472	**2**	Ok	**3**	Ok
330605	**2**	[A2] Warning justify the creation of other modules	**1**	[C1] Global variables lifeform.cpp l.4, scavenger.cpp l.4, Algae.cpp l5, Coral.cpp l.7,8. [C2] shape.h l.15-18
339673	**2**	[A2] Warning justify the creation of other modules	**1**	[C1] Global variables lifeform.cpp l.4, scavenger.cpp l.4, Algae.cpp l5, Coral.cpp l.7,8. [C2] shape.h l.15-18
339966	**1**	[A3] constantes.h et message.h sont inclues dans shape.cc	**3**	Ok
341435	**2**	Ok	**3**	OK MAIS Pénalité dès rendu2: il manque une classe simulation.
344311	**2**	OK	**3**	OK
344324	**2**	OK	**3**	OK
344767	**2**	OK	**3**	OK
346202	**2**	Warning: vous n'avez pas fait attention au makefile: les dépendences des modules sont déclarées plusieurs fois. [A3] lifeform.h est inclus dans shape.h	**3**	[C2] lifeform.h35,41,42 : l'implementation des méthodes dans le fichier d'interface n'est que acceptée pour les constructeurs et les getters de la classe et doit être sur la même ligne.
346300	**1**	[A3]: constantes.h included in shape.cc	**1**	[C2]:lifeform.h40(setListSegments), lifeform.h41 (setListSeg)

347458	**1**	[A3]: constantes.h included in shape.cc	**3**	OK
348518	**2**	[A2] Warning attention a l'ordre des #include ([O11] dans les conventions de programmation). Simulation.cc n'a pas besoin d'avoir des includes deja presents dans son .h.	**2**	[C1] lifeform.cc l.13-15
355624	**2**	OK	**3**	OK, but your static vectors should be defined in source file and not header file
355678	**2**	OK	**0**	[C1]simulation.cc 13,14,15,32,33,34,35,38,39,...
355690	**1**	[A3] shape n'est pas indépendant du modèle	**0**	[C2] lifeform.h:Algue::Algue,Corail::Corail,Scavenger::Scavenger
355754	**1**	Warning: vous n'avez pas fait attention au makefile: les dépendences des modules sont déclarées plusieurs fois. [A3] lifeform.h est inclus dans shape.h	**3**	[C2] lifeform.h35,41,42 : l'implementation des méthodes dans le fichier d'interface n'est que acceptée pour les constructeurs et les getters de la classe et doit être sur la même ligne.
355769	**2**	OK	**3**	OK
355770	**2**	Ok, architectrure correcte et efficace.	**3**	Ok
355786	**2**	Ok	**3**	Ok
355993	**1**	[A3]: constantes.h included in shape.cc	**1**	[C2]:lifeform.h40(setListSegments), lifeform.h41 (setListSeg)
356066	**1**	[A2] inclusion de constantes.h dans shape	**0**	[C2] lifeform.h 41-42, 43-45, 117-119 Warning: 74, 75, 114, 115 L'implémentation des setters ne devrait pas être dans le .h

356068	**2**	OK	**3**	OK
356074	**2**	Parfait, très cohérent!	**3**	Très bien.
356179	**1**	[A2] : "simulation.h" est inclus dans "lifeform.cc"	**0**	[C2] : Shape.h la définition des constructeurs peut être contenue dans l'interface du module uniquement si elle tient sur une seule ligne. Idem pour Algue, Corail et Scavenger dans "lifeform.h". A corriger pour le prochain rendu
356332	**2**	OK mais attention : le mécanisme définissant la valeur de epsil_zero ne pourra plus fonctionner pour le rendu final car ce paramètre devra pouvoir prendre 2 valeurs selon le contexte.	**2**	[C2] lifeform.h ; + warning pour shape.h externaliser la définition des méthodes dans l'implémentation même pour une définition réduite à la liste d'initialisation car cela incite à ne faire aucune vérification.
356447	**2**	OK	**0**	[C1]simulation.cc 13,14,15,32,33,34,35,38,39,...
356459	**0**	[A1] projet.cc instancie des élements de lifeform lui même, lance plusieurs fonctions de simulation, etc. projet.cc doit contenir le strict minimum. [A3] shape.cc inclut constantes.h	**3**	ok mais pourquoi avoir des const et des define qui cohabitent pour shape ?
356483	**2**	OK	**0**	[C1]: lifeform.h (all your classes are defined as struct thereby having the attribute public), [C2] lifeform.h (methods on multiple lines)
356497	**2**	Ok, architectrure correcte et efficace.	**3**	Ok
356509	**2**	OK	**3**	OK
356568	**1**	Très bonne cohérence dans l'architecture, mais attention shape est un module indépendant du projet qui ne doit pas faire apparaitre des mots clés liés à celui-ci. [A3] constantes.h est inclus dans shape.cc.	**3**	Attention au principe d'abstraction: vos fonctions sont trop générales et présentent des effets de bord.

356665	**2**	Bonne architecture	**3**	Très bien. Pour stocker les entités de la simulation tu as à la fois des vector attributs de la classe simulation et à la fois des vector dans le .cc --> met en place une seule des deux solutions
356696	**2**	Bonne architecture	**3**	Très bien
356848	**2**	Warning [A1] vérification d'existence de fichier à déplacer dans le module simulation	**2**	[C2] les constructeurs de Lifeform, Corail et Scavenger doivent être définis dans le .cc s'ils prennent plus d'une ligne. Compacter la définition d'une méthode de cette manière incite à ne pas faire de vérification sur les paramètres, ce qui est une très mauvaise pratique
356960	**2**	OK	**3**	OK
358053	**1**	[A1] projet.cc s'occupe de la lecture de fichier	**3**	Ok
358079	**2**	OK	**3**	OK, but your static vectors should be defined in source file and not header file
358229	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.h"	**3**	Ok
358422	**2**	OK	**2**	[C2] le constructeur d'Algue doit être externalisé
358556	**1**	[A2] lifeform.cc (simulation.h included)	**3**	OK, but your static vectors should be defined in source file and not header file
359309	**1**	[A1] projet.cc doit uniquement inclure simulation comme mentionné à la figure 9a	**3**	OK
359317	**2**	OK	**3**	OK
359350	**1**	Bonne architecture, attention projet est lié au modèle seulement à travers simulation: [A1] shape.h est inclus dans projet.cc, de plus constante.cc n'est pas utile.	**2**	[C2] Les constructeurs sont permis dans l'interface quand ils tiennent dans la même ligne que le prototype: tout vos constructeurs sont trop longs et sont contenus dans les interfaces.

360542	**1**	[A3] constantes.h est inclus dans shape.cc	**3**	OK
360573	**1**	[A2] : "simulation.h" est inclus dans "lifeform.cc"	**0**	[C2] : Shape.h la définition des constructeurs peut être contenue dans l'interface du module uniquement si elle tient sur une seule ligne. Idem pour Algue, Corail et Scavenger dans "lifeform.h". A corriger pour le prochain rendu
360696	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.h"	**3**	Ok
360804	**2**	Ok	**3**	Ok
360955	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.cc"	**3**	Ok
360961	**1**	[A3] constantes.h est inclus dans shape.cc	**3**	OK
360966	**2**	Ok	**3**	Ok
360985	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.cc"	**3**	Ok
361003	**2**	OK	**0**	[C1] lifeform.h (tableu is a non static global variable), [C1/C2] (Segment is a struct but it cannot have method and it was defined in the interface)
361119	**1**	[A3]: constantes.h included in shape.cc	**3**	OK
361158	**2**	Bonne architecture	**3**	Très bien

361167	**2**	Ok, architectrure correcte et efficace.	**3**	Ok
361170	**2**	Ok, architectrure correcte et efficace.	**3**	Ok
361248	**2**	OK	**3**	OK
361295	**2**	[A2] inclusion de constantes.h dans shape	**3**	OK
361373	**2**	Ok	**3**	Ok
361525	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.cc"	**3**	Ok
361740	**2**	OK	**0**	[C1]: lifeform.h (all your classes are defined as struct thereby having the attribute public), [C2] lifeform.h (methods on multiple lines)
361977	**0**	[A1] projet.cc instancie des élements de lifeform lui même, lance plusieurs fonctions de simulation, etc. projet.cc doit contenir le strict minimum. [A3] shape.cc inclut constantes.h	**3**	ok mais pourquoi avoir des const et des define qui cohabitent pour shape ?
362065	**2**	OK	**3**	OK
362182	**2**	OK	**1**	[C2] (2x) les constructeurs de Segment, Scavenger, Corail et Algue doivent être définis dans le .cc s'ils prennent plus d'une ligne. Compacter la définition d'une méthode de cette manière incite à ne pas faire de vérification sur les paramètres, ce qui est une très mauvaise pratique
362197	**2**	ok	**3**	ok

362281	**2**	Ok, architectrure correcte et efficace.	**3**	Ok
362291	**2**	Ok	**3**	Ok
362360	**2**	Bonne architecture	**3**	Ok
362397	**1**	[A2] lifeform.cc (simulation.h included)	**3**	OK, but your static vectors should be defined in source file and not header file
362436	**2**	OK	**1**	[C2] (2x) les constructeurs de Sca, Corail, Algue et Segment doivent être définis dans le .cc s'ils prennent plus d'une ligne. Compacter la définition d'une méthode de cette manière incite à ne pas faire de vérification sur les paramètres, ce qui est une très mauvaise pratique
362702	**2**	OK	**3**	OK
362952	**1**	[A3] shape n'est pas indépendant du modèle	**0**	[C2] lifeform.h:Algue::Algue,Corail::Corail,Scavenger::Scavenger
362974	**2**	Ok	**3**	Ok
362981	**1**	[A3] Constantes.h inclus dans shape.cc	**3**	Bonne structure des classes
363062	**2**	Bonne architecture	**3**	Très bien. Pour stocker les entités de la simulation tu as à la fois des vector attributs de la classe simulation et à la fois des vector dans le .cc --> met en place une seule des deux solutions
363081	**1**	Très bonne cohérence dans l'architecture, mais attention shape est un module indépendant du projet qui ne doit pas faire apparaitre des mots clés liés à celui-ci. [A3] constantes.h est inclus dans shape.cc.	**3**	Attention au principe d'abstraction: vos fonctions sont trop générales et présentent des effets de bord.

363238	**1**	[A3] inclusion de message.h et constante.h dans le module shape	**2**	[C2] la définition de segment::get_extremite doit être externalisée
363255	**1**	[A1] projet.cc doit uniquement inclure simulation comme mentionné à la figure 9a	**3**	OK
363263	**1**	[A3] message.h est inclus dans shape.h.	**3**	OK
363267	**2**	Ok	**3**	OK MAIS Pénalité dès rendu2: il manque une classe simulation.
363275	**2**	Ok	**3**	Ok
363304	**1**	warning: votre architecture doit être validée par l'enseignant. Votre modules erreur n'est pas forcément nécessaire: vous pouvez faire ça dans lifeform. Aussi faites attention à la cohérence de vos includes: errors.cc include lifeform et lifeform.cc include errors.cc. [A3] errors.h inclus dans shape qui doit normalement être independant: c'est lifeform qui fait appel à shape et non l'inverse.	**3**	warning: revoir l'utilisation de l'espace de nom non-nommé.
363429	**1**	[A3] message.h ne doit pas être inclus dans shape.h	**3**	Très bien
363518	**1**	[A3] Constantes.h inclus dans shape.cc	**3**	Bonne structure des classes
363566	**2**	ok	**3**	ok
364037	**1**	[A3] message.h est inclus dans shape.h.	**3**	OK

364053	**1**	Bonne architecture, attention projet est lié au modèle seulement à travers simulation: [A1] shape.h est inclus dans projet.cc, de plus constante.cc n'est pas utile.	**2**	[C2] Les constructeurs sont permis dans l'interface quand ils tiennent dans la même ligne que le prototype: tout vos constructeurs sont trop longs et sont contenus dans les interfaces.
364063	**1**	[A3] message.h est inclus dans shape.h.	**3**	OK
364095	**1**	[A2] inclusion de constantes.h dans shape	**0**	[C2] lifeform.h 41-42, 43-45, 117-119 Warning: 74, 75, 114, 115 L'implémentation des setters ne devrait pas être dans le .h
364173	**2**	OK	**3**	OK
371921	**2**	OK	**1**	[C2] (2x) les constructeurs de Sca, Corail, Algue et Segment doivent être définis dans le .cc s'ils prennent plus d'une ligne. Compacter la définition d'une méthode de cette manière incite à ne pas faire de vérification sur les paramètres, ce qui est une très mauvaise pratique
371946	**1**	[A3] inclusion de constantes.h dans le module shape	**2**	[C2] les constructeurs de Lifeform, Coral et Scavenger doivent être externalisés
371969	**2**	OK	**3**	OK
371989	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.cc"	**3**	Ok
372017	**2**	warning shape.h:8 enum trop spécifique pour que cela soit dans le module shape, je conseille d'utiliser un booléen pour choisir si oui ou non utiliser epsilon zero	**3**	OK
372018	**2**	OK	**3**	OK

372048	**1**	[A3] constante.h est inclut dans shape.h	**3**	OK
372051	**1**	[A3] constante.h est inclus dans shape.h, shape.cc	**3**	OK
372071	**2**	Bonne architecture	**3**	OK. Si tu utilises une fonction uniquement dans le .cc de ton module met son prototype dans le .cc et pas dans le .h (exemple : "age_verif" du module lifeform)
372082	**1**	warning: votre architecture doit être validée par l'enseignant pour l'ajout du module reading. [A3] constexpr.h est inclus dans shape.h	**3**	OK
372083	**1**	warning: votre architecture doit être validée par l'enseignant. Votre modules erreur n'est pas forcément nécessaire: vous pouvez faire ça dans lifeform. Aussi faites attention à la cohérence de vos includes: errors.cc include lifeform et lifeform.cc include errors.cc. [A3] errors.h inclus dans shape qui doit normalement être independant: c'est lifeform qui fait appel à shape et non l'inverse.	**3**	warning: revoir l'utilisation de l'espace de nom non-nommé.
372118	**1**	[A3] message.h est inclus dans shape.h.	**3**	OK
372124	**2**	OK	**3**	OK
372178	**2**	OK	**3**	OK

372192	**2**	OK	**3**	OK
372212	**2**	Ok, bonne architecture	**3**	Bonne structure des classes
372226	**2**	Ok	**3**	Attention, la façon dont vous mettez en oeuvre les tests sur les données de lecture met en péril votre encapsulation dans la mesure ou les tests sont demandés par la simulation une fois que toutes les entités ont été crées. Vous auriez probablement meilleur temps de déléguer cette tâche au module lifeform
372331	**2**	Warning: vous avez inclu shape.h dans les dépendances du module message par erreur dans le makefile	**3**	OK
372387	**1**	[A3] constante.h ne doit pas être inclus dans shape.h	**2**	[C2] shape.h : Externalise la définition du constructeur car elle ne tient pas sur une ligne.
372414	**2**	Warning: vous avez inclu shape.h dans les dépendances du module message par erreur dans le makefile	**3**	OK
372417	**2**	Ok	**3**	OK MAIS Pénalité dès rendu2: il manque une classe simulation.
372427	**2**	Ok	**3**	Ok
372454	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.cc"	**3**	Ok
372460	**2**	Bonne architecture. Attention dans lifeform.h ligne 5 à ton #define.	**0**	[C2] lifeform.h et shape.h : Externalise les définitions de tes méthodes et constructeurs qui ne tiennent pas sur une seule ligne. Si tu utilises une fonction uniquement dans le .cc de ton module met son prototype dans le .cc et pas dans le .h (exemple : "detect_pos_age" du module lifeform)
372468	**2**	Bonne architecture de projet	**3**	[C1] Warning pas de label static simulation.cc l.17-20 mm si vous avez deja mis le label dans le .h

372474	**2**	ok	**0**	[C2] simulation.h:Simulation::Simulation lifeform.h:corail::corail,scavenger::scavenger
372570	**2**	OK	**3**	OK
372602	**2**	OK	**2**	[C2] shape.h (externalization of methods apart from getters and constructors that don't fit onto oneline)
372625	**1**	[A3] shape.cc n'est pas indépendant du modèle	**3**	ok
372633	**2**	OK	**2**	[C2] shape.h (externalization of methods apart from getters and constructors that don't fit onto oneline)
372689	**2**	Ok	**3**	Ok
372757	**2**	OK	**0**	[C1] Simulation::compteur, Simulation::total, Simulation::i attributs de classe static publics, Simulation::list_alg, Simulation::list_cor, Simulation::list_sca attributs publics
372791	**1**	warning: "pragma once" n'est pas standard en C++. [A3] constantes.h est inclus dans shape.h et shape.cc	**3**	ok
372830	**2**	OK	**2**	[C1] lifeform.h:Algue::nb_seg
372924	**2**	Bonne architecture	**0**	[C2] : simulation.h et lifeform.h : externalise les définitions des constructeurs qui ne tiennent pas sur une seule ligne
373043	**2**	OK	**3**	OK

373083	**2**	Ok	**3**	Ok
373154	**2**	OK	**2**	[C2] le constructeur d'Algue doit être externalisé
373274	**2**	Ok	**3**	OK MAIS Pénalité dès rendu2: il manque une classe simulation.
373282	**2**	OK	**3**	OK
373338	**2**	Bonne architecture	**3**	OK. Si tu utilises une fonction uniquement dans le .cc de ton module met son prototype dans le .cc et pas dans le .h (exemple : "age_verif" du module lifeform)
373355	**2**	OK	**3**	OK
373398	**2**	OK	**3**	OK
373412	**2**	OK	**3**	OK
373418	**2**	ok	**3**	ok
373427	**2**	warning shape.h:8 enum trop spécifique pour que cela soit dans le module shape, je conseille d'utiliser un booléen pour choisir si oui ou non utiliser epsilon zero	**3**	OK
373443	**2**	Très bonne architecture, bon respect des includes et de la cohérence des interdépendances entre modules.	**3**	Très bien.
373447	**1**	[A3] message.h ne doit pas être inclus dans shape.h	**3**	OK. Si tu utilises une fonction uniquement dans le .cc de ton module met son prototype dans le .cc et pas dans le .h (exemple : "simu" du module simulation)

373465	**2**	OK	**3**	OK, but in shape.cc, Segment has a method while being a struct, convert it to class
373482	**2**	Bonne architecture	**1**	[C2] lifeform.h, shape.h : externalise les définitions des méthodes qui ne tiennent pas sur une ligne. Warning : évite au maximum les prototypes de fonctions dans tes .h si tu as des classes --> fais plutôt des méthodes public ou private selon si la fonction peut être accédée par d'autres modules ou non. Tu as une classe simulation avec des vector private mais tu ne crées pas d'objet simulation et remet ces vector dans ton .cc --> fais un objet simulation et implémente seulement l'une ou l'autre manière de stocker les objets de la simulation.
373549	**2**	Très bonne architecture.	**3**	Très bien.
373768	**2**	ok	**3**	ok
373799	**2**	OK mais attention : le mécanisme définissant la valeur de epsil_zero ne pourra plus fonctionner pour le rendu final car ce paramètre devra pouvoir prendre 2 valeurs selon le contexte.	**2**	[C2] lifeform.h ; + warning pour shape.h externaliser la définition des méthodes dans l'implémentation même pour une définition réduite à la liste d'initialisation car cela incite à ne faire aucune vérification.
373831	**1**	warning: votre architecture doit être validée par l'enseignant. Votre modules in_out n'est pas forcément nécessaire et crée des repetitions, essayez de regrouper les choses: vous avez 3 fonctions dans trois modules differents qui servent à créer une algue ce qui crée des incohérences. [A1] projet appelle une fonction de in_out "val_ord": projet n'a accès au modèle que à travers simulation.	**1**	[C1] variable globale dans lifeform.cc: l.8. Vous avez trop de classes et certaines se répètent ce qui crée la confusion.
373863	**2**	ok	**0**	[C2] simulation.h:Simulation::Simulation lifeform.h:corail::corail,scavenger::scavenger
373877	**1**	Bon respect de l'architecture, mais attention shape est un module indépendant du projet qui ne doit pas faire apparaitre des mots clés liés à celui-ci. [A3] constantes.h est inclus dans shape.cc.	**3**	Attention au principe d'abstraction: vos fonctions sont trop générales et présentent des effets de bord.

373990	**2**	Bonne architecture	**1**	[C2] lifeform.h, shape.h : externalise les définitions des méthodes qui ne tiennent pas sur une ligne. Warning : évite au maximum les prototypes de fonctions dans tes .h si tu as des classes --> fais plutôt des méthodes public ou private selon si la fonction peut être accédée par d'autres modules ou non. Tu as une classe simulation avec des vector private mais tu ne crées pas d'objet simulation et remet ces vector dans ton .cc --> fais un objet simulation et implémente seulement l'une ou l'autre manière de stocker les objets de la simulation.
373994	**1**	warning: votre architecture doit être validée par l'enseignant. la répartition des fonctionalités "lecture" et "algo" n'est pas franchement nécessaire ; ces modules peuvent être fusionnés avec simulation. préciser où se trouve le module seg : est-il dans le Modèle ? [A3] constantes.h est inclus dans shape.h.	**3**	OK MAIS Pénalité dès rendu2: il manque une classe simulation.
374000	**1**	[A1] project.cc ne peut pas inclure autre chose que simulation.cc pour ce rendu ; WARNING car rien de shape n'est utilisé, mais supprimez la ligne de l'include pour le prochain rendu [A3] shape doit être indépendant du modèle	**1**	[C2] lifeform.h:Corail::setSegment shape.h:Segment::Segment
374009	**1**	[A3] Attention, shape doit être indépendant de constantes.h (7.3.1)	**3**	OK
374108	**2**	Ok, architectrure correcte et efficace.	**2**	[C2]lifeform.h : la définition des constructeurs peuvent être contenus dans l'interface du module uniquement si elle tient sur une seule ligne.
374239	**2**	Ok, architectrure correcte et efficace.	**2**	[C2]lifeform.h : la définition des constructeurs peuvent être contenus dans l'interface du module uniquement si elle tient sur une seule ligne.
374440	**1**	warning: votre architecture doit être validée par l'enseignant. Votre modules in_out n'est pas forcément nécessaire et crée des repetitions, essayez de regrouper les choses: vous avez 3 fonctions dans trois modules differents qui servent à créer une algue ce qui crée des incohérences. [A1] projet appelle une fonction de in_out "val_ord": projet n'a accès au modèle que à travers simulation.	**1**	[C1] variable globale dans lifeform.cc: l.8. Vous avez trop de classes et certaines se répètent ce qui crée la confusion.

374441	**2**	Ok	**3**	Ok
374657	**2**	ok	**3**	ok
374686	**0**	Attention au module projet: projet.h n'est pas nécessaire=> aucun module ne l'inclus, de plus projet est lié au modèle seulement à travers simulation: [A1] message.h est inclus dans projet.cc. [A3] constante.h est inclus dans shape.cc. Il faut inclure le moins possible: surtout dans les interfaces (lifeform.h include constante.h et message.h sans raison).	**2**	[C2] Les constructeurs sont permis dans l'interface quand ils tiennent dans la même ligne que le prototype: tout vos constructeurs sont trop longs et sont contenus dans les interfaces.
374686	**0**	Attention au module projet: projet.h n'est pas nécessaire=> aucun module ne l'inclus, de plus projet est lié au modèle seulement à travers simulation: [A1] message.h est inclus dans projet.cc. [A3] constante.h est inclus dans shape.cc. Il faut inclure le moins possible: surtout dans les interfaces (lifeform.h include constante.h et message.h sans raison).	**2**	[C2] Les constructeurs sont permis dans l'interface quand ils tiennent dans la même ligne que le prototype: tout vos constructeurs sont trop longs et sont contenus dans les interfaces.
374966	**0**	[A1], [A2], [A3] il y a que le module projet	**0**	[C1] projet.cpp:91-98, etc.. [C2] projet.cpp:49,66
375085	**2**	OK	**3**	OK
375108	**1**	[A3] message.h ne doit pas être inclus dans shape.h	**3**	Très bien
375121	**2**	OK	**3**	OK

375202	**2**	OK	**3**	OK
375206	**1**	[A1] project.cc ne peut pas inclure autre chose que simulation.cc pour ce rendu ; WARNING car rien de lifeform n'est utilisé, mais supprimez la ligne de l'include pour le prochain rendu [A3] shape.cc n'est pas indépendant du modèle	**3**	ok
375214	**2**	OK	**3**	OK
375236	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.cc"	**3**	Ok
375353	**2**	OK	**3**	OK, but in shape.cc, Segment has a method while being a struct, convert it to class
375361	**2**	OK	**3**	OK
375362	**2**	OK	**3**	OK
375364	**2**	Ok, architectrure correcte et efficace.	**2**	[C2]simulation.h35 : la définition des constructeurs peuvent être contenus dans l'interface du module uniquement si elle tient sur une seule ligne.
375390	**1**	[A3] inclusion de constantes.h dans le module shape	**2**	[C2] les constructeurs de Lifeform, Coral et Scavenger doivent être externalisés
375399	**2**	OK	**3**	OK
375427	**2**	ok	**3**	ok

375454	**2**	Bonne architecture de projet	**3**	Bonne structure des classes
375632	**2**	OK	**3**	OK
376115	**2**	OK	**3**	OK
376265	**2**	Bonne architecture de projet	**3**	Bonne structure des classes
376501	**2**	OK	**0**	[C1] Simulation::compteur, Simulation::total, Simulation::i attributs de classe static publics, Simulation::list_alg, Simulation::list_cor, Simulation::list_sca attributs publics
376698	**2**	ok	**3**	ok
376735	**1**	[A1] project.cc ne peut pas inclure autre chose que simulation.cc pour ce rendu ; WARNING car rien de lifeform n'est utilisé, mais supprimez la ligne de l'include pour le prochain rendu [A3] shape.cc n'est pas indépendant du modèle	**3**	ok
376800	**2**	Bonne architecture de projet	**3**	[C1] Warning pas de label static simulation.cc l.17-20 mm si vous avez deja mis le label dans le .h
376866	**2**	[A2] Warning on #include orders ([O11] in conventions de programmation). Simulation.cc doens't need includes that are already in its .h	**3**	Good class structure
377004	**1**	[A3] constantes.cc est inclu dans shape.cc	**3**	OK. Bon travail sur la modularisation, pour mieux encapsuler, je pense que les fonctions dans lifeform (validerScavenger, ...) pourrait être associé à leur classe respectives.

377096	**2**	OK	**3**	OK
377272	**2**	Ok	**3**	Ok
377774	**1**	warning: votre architecture doit être validée par l'enseignant pour l'ajout du module reading. [A3] constexpr.h est inclus dans shape.h	**3**	OK
378145	**2**	[A2] Warning on #include orders ([O11] in conventions de programmation). Simulation.cc doens't need includes that are already in its .h	**3**	Good class structure
378335	**2**	OK	**3**	OK
378336	**2**	OK	**3**	Ok
378408	**1**	[A3] constantes.h est inclu dans shape.h	**3**	Ok
378490	**1**	[A1] projet.cc s'occupe de la lecture de fichier	**3**	Ok
378517	**2**	OK	**0**	[C1] lifeform.h (tableu is a non static global variable), [C1/C2] (Segment is a struct but it cannot have method and it was defined in the interface)
378523	**2**	Ok	**3**	Ok
378533	**2**	OK	**3**	OK
378614	**2**	OK	**3**	OK

378660	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.cc"	**3**	Ok
378744	**2**	OK	**3**	OK
378782	**2**	Ok, architectrure correcte et efficace.	**3**	Warning : Les static vectors qui contiennent vos entités sont assez dangereux si ils sont utilisés en dehors d'un contexte d'allocation dynamique, il faudra justifier ça dans le rapport du prochain rendu
378979	**1**	warning: "pragma once" n'est pas standard en C++. [A3] constantes.h est inclus dans shape.h et shape.cc	**3**	ok
379052	**2**	OK	**3**	OK
379114	**2**	Très bonne architecture.	**3**	Très bien.
379158	**2**	Parfait, très cohérent!	**3**	Très bien.
379194	**2**	Bonne architecture, cohérence dans les includes et dans les liens entre modules	**3**	Attention au principe d'abstraction: vos fonctions sont trop générales et présentent des effets de bord: faites plusieurs petites fonctions erreurs.
379270	**2**	Bonne architecture	**0**	[C2] : simulation.h et lifeform.h : externalise les définitions des constructeurs qui ne tiennent pas sur une seule ligne
379329	**2**	OK	**3**	OK
379343	**1**	[A1] message et shape inclus dans projet.cc. [A2] Warning include dans le .cc deja presents dans le .h	**3**	Bonne structure des classes

379344	**1**	[A3] message.h ne doit pas être inclus dans shape.h	**3**	OK. Si tu utilises une fonction uniquement dans le .cc de ton module met son prototype dans le .cc et pas dans le .h (exemple : "simu" du module simulation)
379375	**1**	[A1] projet.cc doit uniquement inclure simulation comme mentionné à la figure 9a	**2**	[C2] lifeform.h get_bras(),id_cible,get_sts()
379382	**2**	ok	**0**	[C1] lifeform.h:Algues::x1,Algues::y1,Algues::age1,..
379386	**2**	Bonne architecture, cohérence dans les includes et dans les liens entre modules	**3**	Attention au principe d'abstraction: vos fonctions sont trop générales et présentent des effets de bord: faites plusieurs petites fonctions erreurs.
379509	**2**	OK	**3**	OK
379550	**1**	[A3] constantes.h est inclu dans shape.h	**3**	Ok
379654	**1**	[A2] inclusion de constantes.h dans shape	**0**	[C2] lifeform.h 25,28-30,48-49,54-56,98-99,104-16 shape.h: 25-30
379659	**1**	[A3] constantes.cc est inclu dans shape.cc	**3**	OK. Bon travail sur la modularisation, pour mieux encapsuler, je pense que les fonctions dans lifeform (validerScavenger, ...) pourrait être associé à leur classe respectives.
379660	**1**	[A1] projet.cc doit uniquement inclure simulation comme mentionné à la figure 9a	**2**	[C2] lifeform.h get_bras(),id_cible,get_sts()
379664	**1**	[A3] : Le module "constante.h" ne doit pas etre inclus dans "shape.cc"	**3**	Ok
379714	**1**	[A1] project.cc ne peut pas inclure autre chose que simulation.cc pour ce rendu ; WARNING car rien de shape n'est utilisé, mais supprimez la ligne de l'include pour le prochain rendu [A3] shape doit être indépendant du modèle	**1**	[C2] lifeform.h:Corail::setSegment shape.h:Segment::Segment

379723	**2**	Ok	**3**	Ok
379734	**2**	Ok	**3**	Ok
379816	**0**	[A1], [A2], [A3] il y a que le module projet	**0**	[C1] projet.cpp:91-98, etc.. [C2] projet.cpp:49,66
379827	**2**	Warning [A1] inclusion inutile de shape.h dans projet.cc.	**3**	OK
379847	**2**	Good architecture	**3**	Good structure overal, but Warning on the size of lifeform.cc, maybe try to put some fonctions in simulations as it is almost empty and lifeform has all the functions
379874	**1**	[A3] constantes.h est inclu dans shape.h	**3**	Ok
379882	**2**	OK	**1**	[C2] (2x) les constructeurs de Segment, Scavenger, Corail et Algue doivent être définis dans le .cc s'ils prennent plus d'une ligne. Compacter la définition d'une méthode de cette manière incite à ne pas faire de vérification sur les paramètres, ce qui est une très mauvaise pratique
379889	**2**	OK	**3**	OK
379895	**2**	Très bonne architecture, seulement Il faut inclure le moins d'interfaces possible: surtout dans les interfaces (lifeform.h include constante.h sans raison).	**3**	Très bien.
379911	**2**	Ok, bonne architecture	**3**	Bonne structure des classes
379915	**1**	[A1] : le module projet doit uniquement gérer l'argument de la lig,e de commande. Il doit uniquement appeler une méthode du module Simulation.	**3**	WARNING: vous devez avoir un module projet.cc et Simulation (pas io et prog). De plus il manque une classe simulation.

379988	**2**	OK	**3**	OK
379991	**2**	OK	**3**	OK
380052	**2**	Ok, architectrure correcte et efficace.	**3**	Ok
380080	**2**	ok	**3**	ok
380082	**2**	Ok, architectrure correcte et efficace.	**3**	Ok
380090	**1**	[A3] constantes.h est inclus dans shape.h.	**2**	[C2] lifeform.h61-63
380097	**2**	OK	**3**	Ok
380105	**2**	[A2] Warning simulation.h n'est pas inclut dans le .cc	**3**	Warning [C2] shape.cc l.10-12 ces prototypes doivent se trouver dans le .h
380137	**1**	[A3] constantes.h est inclus dans shape.h.	**2**	[C2] lifeform.h61-63
380154	**2**	[A2] Warning attention a l'ordre des #include ([O11] dans les conventions de programmation). Simulation.cc n'a pas besoin d'avoir des includes deja presents dans son .h.	**2**	[C1] lifeform.cc l.13-15
380200	**2**	OK	**2**	[C1] lifeform.h:Algue::nb_seg
380211	**1**	[A3] message.h est inclus dans shape.h	**2**	[C2] les méthodes add de la classe simulation

380226	**2**	OK	**3**	OK
380234	**1**	Bon respect de l'architecture, mais attention shape est un module indépendant du projet qui ne doit pas faire apparaitre des mots clés liés à celui-ci. [A3] constantes.h est inclus dans shape.cc.	**3**	Attention au principe d'abstraction: vos fonctions sont trop générales et présentent des effets de bord.
380279	**1**	[A1] : le module projet doit uniquement gérer l'argument de la lig,e de commande. Il doit uniquement appeler une méthode du module Simulation.	**3**	WARNING: vous devez avoir un module projet.cc et Simulation (pas io et prog). De plus il manque une classe simulation.
380349	**1**	[A3] constantes.h et message.h sont inclues dans shape.cc	**3**	Ok
380446				
380459	**2**	[A2] Warning simulation.h n'est pas inclut dans le .cc	**3**	Warning [C2] shape.cc l.10-12 ces prototypes doivent se trouver dans le .h
380517	**1**	[A3] Attention, shape doit être indépendant de constantes.h (7.3.1)	**3**	OK
380538	**2**	OK	**3**	OK
380606	**1**	[A3] inclusion de message.h et constante.h dans le module shape	**2**	[C2] la définition de segment::get_extremite doit être externalisée
380610	**2**	Bonne architecture. Attention dans lifeform.h ligne 5 à ton #define.	**0**	[C2] lifeform.h et shape.h : Externalise les définitions de tes méthodes et constructeurs qui ne tiennent pas sur une seule ligne. Si tu utilises une fonction uniquement dans le .cc de ton module met son prototype dans le .cc et pas dans le .h (exemple : "detect_pos_age" du module lifeform)
380642	**2**	Ok, architectrure correcte et efficace.	**2**	[C2]simulation.h35 : la définition des constructeurs peuvent être contenus dans l'interface du module uniquement si elle tient sur une seule ligne.

380661	**2**	OK	**3**	OK
380664	**2**	Bonne architecture	**3**	Très bien
380673	**2**	OK	**1**	[C2] définition du constructeur de Segment à externaliser. [C2] Simulation: getters à externaliser s'ils ne tiennent pas sur une ligne
380682	**2**	Bonne architecture	**3**	OK. Si tu utilises une fonction uniquement dans le .cc de ton module met son prototype dans le .cc et pas dans le .h (exemple : "in_wolrd" du module lifeform)
380756	**2**	OK	**3**	OK
380773	**2**	[A2] inclusion de constantes.h dans shape	**3**	OK
380788	**2**	Bonne architecture	**3**	Ok
380883	**1**	[A3] constante.h est inclus dans shape.h, shape.cc	**3**	OK
380931	**1**	[A2] inclusion de constantes.h dans shape	**0**	[C2] lifeform.h 25,28-30,48-49,54-56,98-99,104-16 shape.h: 25-30
380963	**1**	[A1] module projet absent	**2**	[C2] simulation: la définition des méthodes n'est pas externalisée en dehors de la déclaration de la classe
380988	**2**	Warning [A1] vérification d'existence de fichier à déplacer dans le module simulation	**2**	[C2] les constructeurs de Lifeform, Corail et Scavenger doivent être définis dans le .cc s'ils prennent plus d'une ligne. Compacter la définition d'une méthode de cette manière incite à ne pas faire de vérification sur les paramètres, ce qui est une très mauvaise pratique

381001	**1**	[A1] message et shape inclus dans projet.cc. [A2] Warning include dans le .cc deja presents dans le .h	**3**	Bonne structure des classes
381013	**2**	OK	**1**	[C2] définition du constructeur de Segment à externaliser. [C2] Simulation: getters à externaliser s'ils ne tiennent pas sur une ligne
381098	**1**	[A3] constantes.h est inclu dans shape.h	**3**	Ok
381128	**2**	Ok	**3**	Ok
381136	**2**	Good architecture	**3**	Good structure overal, but Warning on the size of lifeform.cc, maybe try to put some fonctions in simulations as it is almost empty and lifeform has all the functions
381157	**2**	Ok	**3**	Attention, la façon dont vous mettez en oeuvre les tests sur les données de lecture met en péril votre encapsulation dans la mesure ou les tests sont demandés par la simulation une fois que toutes les entités ont été crées. Vous auriez probablement meilleur temps de déléguer cette tâche au module lifeform
381162	**1**	[A3] constante.h est inclut dans shape.h	**3**	OK
381171	**2**	OK	**3**	OK
381175	**1**	[A3] constante.h ne doit pas être inclus dans shape.h	**2**	[C2] shape.h : Externalise la définition du constructeur car elle ne tient pas sur une ligne
381229	**2**	OK	**3**	OK
381243	**1**	[A3] message.h est inclus dans shape.h	**2**	[C2] les méthodes add de la classe simulation

381357	**2**	Ok, architectrure correcte et efficace.	**3**	Ok
381391	**2**	OK	**3**	OK
381393	**2**	Warning [A1] inclusion inutile de shape.h dans projet.cc.	**3**	OK
381402	**2**	Bonne architecture	**3**	OK. Si tu utilises une fonction uniquement dans le .cc de ton module met son prototype dans le .cc et pas dans le .h (exemple : "age_verif" du module lifeform)
381646	**1**	warning: votre architecture doit être validée par l'enseignant. la répartition des fonctionalités "lecture" et "algo" n'est pas franchement nécessaire ; ces modules peuvent être fusionnés avec simulation. préciser où se trouve le module seg : est-il dans le Modèle ? [A3] constantes.h est inclus dans shape.h.	**3**	OK MAIS Pénalité dès rendu2: il manque une classe simulation.
381765	**1**	[A3] shape.cc n'est pas indépendant du modèle	**3**	ok
381793	**2**	Ok	**3**	Ok
381905	**2**	Bonne architecture	**3**	Très bien
381925	**2**	Ok, architectrure correcte et efficace.	**3**	Warning : Les static vectors qui contiennent vos entités sont assez dangereux si ils sont utilisés en dehors d'un contexte d'allocation dynamique, il faudra justifier ça dans le rapport du prochain rendu
381964	**2**	Très bonne architecture, bon respect des includes et de la cohérence des interdépendances entre modules.	**3**	Très bien.

381993	**2**	ok	**0**	[C1] lifeform.h:Algues::x1,Algues::y1,Algues::age1,..
382029	**2**	Très bonne architecture, seulement Il faut inclure le moins d'interfaces possible: surtout dans les interfaces (lifeform.h include constante.h sans raison).	**3**	Très bien.