T-Rex Environment Testing Instructions

📋 Prerequisites

Before testing, ensure you have:

1. **Python packages installed:**

bash

```
pip install gymnasium stable-baselines3 numpy pillow
```

2. **Project structure:**

```
your_project/
├── server.py              # HTTP server (NEW)
├── test_env.py            # Test script (NEW)
├── Env_Dino.py            # Corrected environment
├── socket_test.py         # WebSocket bridge
└── t-rex-runner/          # Game directory
    ├── index.html
    ├── index.js
    └── ... (other game files)
```

🚀 How to Run Tests

Step 1: Start the HTTP Server

Open a terminal and run:

bash

```
python server.py
```

You should see:

```
============================================================
T-REX GAME HTTP SERVER
============================================================
📁 Serving files from: /path/to/t-rex-runner
📄 Files in directory:
  - index.html
  - index.js
  ...

============================================================
🌐 Server running at http://localhost:8000
👉 Open http://localhost:8000/index.html in your browser
============================================================
```

⚠️ **Keep this terminal running!**

Step 2: Open the Game in Browser

1. Open your web browser
2. Navigate to: `http://localhost:8000/index.html`
3. You should see the T-Rex game
4. Wait for "WebSocket connected - Browser ready!" in browser console (F12)

Step 3: Run the Tests

Open a **new terminal** (keep the server running) and run:

bash

```
python test_env.py
```

Press Enter when prompted after confirming the server is running.

📊 Test Suites

The test script runs three comprehensive test suites:

1. Basic Functionality Test

- Environment creation
- Reset functionality
- Action space validation
- Observation space validation
- Episode completion

2. Observation Consistency Test

- Observation structure validation
- Frame shape and dtype checks
- Game state completeness
- Value range validation
- Cross-step consistency

3. Stable-Baselines3 Compatibility Test

- Full SB3 `check_env()` validation
- Action/observation space compatibility
- Reward, done, info format validation
- Ensures compatibility with PPO, DQN, A2C, etc.

✅ Expected Output

If everything works correctly, you'll see:

📋

```
============================================================
TEST SUMMARY
============================================================
Basic Functionality..... ✓ PASSED
Observation Consistency..... ✓ PASSED
SB3 Compatibility..... ✓ PASSED
============================================================
🎉 ALL TESTS PASSED! Environment is ready for training!

You can now use this environment with SB3 algorithms like:
  - PPO
  - DQN
  - A2C
  - etc.
============================================================
```

🐛 Troubleshooting

Error: "Connection refused" or "WebSocket connection failed"

Solution: Make sure:

1. HTTP server is running (`python server.py`)
2. Game is open in browser
3. Browser console shows "WebSocket connected"

Error: "Port 8000 already in use"

Solution:

python

```
# Edit server.py and change:
PORT = 8001  # or any other free port
```

Error: "Tainted canvas" or CORS error

Solution:

- Always use the provided `server.py`
- Don't open `index.html` directly as a file (file://)
- Must use http://localhost:8000/index.html

Error: "Game directory not found"

Solution:

bash

```
# Check your directory structure
ls -la t-rex-runner/

# Make sure index.html exists
ls t-rex-runner/index.html
```

Browser shows game but tests fail

Solution:

1. Open browser console (F12)
2. Look for WebSocket connection message
3. Check for JavaScript errors
4. Refresh the page
5. Wait 2-3 seconds before running tests

🔍 Manual Testing

If automated tests fail, try manual testing:

python

```python
from Env_Dino import Env_Dino
import time

# Create environment
env = Env_Dino()
time.sleep(1)  # Wait for connection

# Reset
obs, info = env.reset()
print(f"Observation shape: {obs['frames'].shape}")
print(f"Score: {info['score']}")

# Take some actions
for i in range(5):
    obs, reward, done, truncated, info = env.step(1)  # Jump
    print(f"Step {i}: reward={reward}, score={info['score']}")
    time.sleep(0.1)

env.close()
```

📈 Next Steps After Passing Tests

Once all tests pass, you can train RL agents:


python

```
from stable_baselines3 import PPO
from Env_Dino import Env_Dino

# Create environment
env = Env_Dino()

# Create PPO agent
model = PPO("MultiInputPolicy", env, verbose=1)

# Train
model.learn(total_timesteps=100_000)

# Save
model.save("trex_ppo")

# Test
obs, _ = env.reset()
for _ in range(1000):
    action, _ = model.predict(obs, deterministic=True)
    obs, reward, terminated, truncated, info = env.step(action)
    if terminated or truncated:
        obs, _ = env.reset()
```

Notes

- Tests may take 2-5 minutes to complete
- The environment needs ~30ms per step due to game rendering
- Frame stacking uses the last 4 frames (240ms of history)
- Maximum score depends on how long the agent survives

Getting Help

If tests still fail after troubleshooting:

1. Check browser console for errors (F12)
2. Check server terminal for connection logs
3. Verify all files are in the correct location
4. Ensure Python packages are up to date:

bash

```
pip install --upgrade gymnasium stable-baselines3
```

✨ Success Criteria

Your environment is ready when:

- ✅ All three test suites pass
- ✅ No warnings from `check_env()`
- ✅ Observations are consistent
- ✅ Actions produce expected results
- ✅ Episodes terminate correctly